IMS/ESA Version 5 Performance Guide

December 1997
Chapter 7. Understanding Problems with the I/O Subsystem

7.1 I/O Problem Identification Chart
7.2 Channels
  7.2.1 Channel Busy Chart
  7.2.2 Channel Imbalance
  7.2.3 Channel I/O Activity
7.3 Device Activity
  7.3.1 Device I/O Activity
  7.3.2 Data Set Placement and Allocation
  7.3.3 Shared DASD
  7.3.4 Control Unit, String, Device Busy
7.4 Miscellaneous
  7.4.1 Shared DASD Chart
  7.4.2 Shared DASD: Control Units and Devices
  7.4.3 Private DASD
  7.4.4 VSAM Key Sequenced Data Set CI/CA Splits
  7.4.5 Monitoring Procedures
7.5 IMS Data Sets Chart
  7.5.1 Mean I/O IWAIT Times
  7.5.2 Dispatching Priority and I/O IWAIT Times
  7.5.3 I/O Priorities
  7.5.4 IMS Data Set Placement
7.6 IMS Message Regions

Chapter 8. Understanding IMS Use of Real Storage and Pools

8.1 Storage/Pools Problem Identification Chart
8.2 Page Faulting
8.3 System Paging
  8.3.1 IMS Dependent-Region Swapping
  8.3.2 IMS Batch DL/I Region Swapping
  8.3.3 Paging Data Sets
  8.3.4 MVS Fix List
  8.3.5 IMS Batch DL/I BLDL Entries
8.4 IMS Paging
  8.4.1 Storage Isolation of the IMS Control Region
  8.4.2 Storage Isolation of the DL/I Subordinate Address Space
  8.4.3 Storage Isolation of CSA
  8.4.4 IMS Page Fixing
8.5 IMS Control Blocks
  8.5.1 Overview
  8.5.2 IMS Control Blocks Summary
  8.5.3 Save Area Prefixes
8.6 IMS System Checkpoints
  8.6.1 Checkpoint Logging
  8.6.2 Monitoring Checkpoints
  8.6.3 Monitoring System Paging at Checkpoint Time
9.10.4 EMH Buffer Monitoring and Tuning ............................................  184
9.11 Message Format Pool ..............................................................  184
  9.11.1 MFS Block Fetch ............................................................  184
  9.11.2 Prefetch and Immediate Fetch ............................................  185
  9.11.3 MFS Buffer Pool Queues ...................................................  186
  9.11.4 Fetch Request Elements ...................................................  186
  9.11.5 MFS Pool .................................................................  186
  9.11.6 MFS I/O Performance .......................................................  187
  9.11.7 MFS Performance Considerations ........................................  188
  9.11.8 Test MFS .........................................................................  188
9.12 Scheduling Pools ........................................................................  188
9.13 PSB Pool ...................................................................................  189
  9.13.1 PSB Pools .........................................................................  190
  9.13.2 Resident PSBs and Intent Lists .............................................  191
  9.13.3 PSB Scheduling .................................................................  192
  9.13.4 PSB Pool Space Management ...............................................  192
  9.13.5 Dynamic PSB Option ..........................................................  193
  9.13.6 PSB Pool Monitoring ..........................................................  193
  9.13.7 PSB Pool Tuning .................................................................  193
9.14 Extended PCB Pool .................................................................  194
9.15 PSB Work Pool ..........................................................................  194
9.16 DMB Pool .................................................................................  196
9.17 Database Work Pool .................................................................  197
9.18 Exclusive Control Enqueue/Dequeue Pool ....................................  197
  9.18.1 Overview of Program Isolation ENQ/DEQ ...............................  197
  9.18.2 Control Block Structure .......................................................  197
  9.18.3 Monitoring PI Enq/Deq Pool Usage ........................................  199
9.19 IRLM .........................................................................................  199
  9.19.1 IRLM Dispatching Priority ...................................................  200
  9.19.2 Execution Parameters ..........................................................  200
  9.19.3 IRLM Monitoring .................................................................  200
9.20 Log Buffer Pool .........................................................................  200
  9.20.1 Monitoring Logging Performance ..........................................  201
  9.20.2 Reducing the Volume of Log Data .........................................  201
  9.20.3 Log Buffer Tuning .................................................................  202
  9.20.4 Logging Considerations When Using Fast Path .......................  202
  9.20.5 Monitoring Fast Path CI Logging ...........................................  203
9.21 IMS Monitor Buffer Pool ...........................................................  203
9.22 Automated Operator Interface Pool .............................................  204
9.23 Work Area or General Pool .........................................................  204
9.24 Fast Path Work Pool ..................................................................  205
9.25 Communications External Subsystem Pool ....................................  205

Chapter 10. Scheduling and Dependent Region Considerations ..............  207
10.1 Dependent Region Problem Identification Chart .............................  207
10.2 Program Scheduling ....................................................................  207
  10.2.1 Class Scheduling ................................................................  208
  10.2.2 Scheduling Classes and the MVS Work Load Manager .............  209
  10.2.3 Wait-for-Input Transactions ...............................................  209
  10.2.4 Pseudo-WFI Transactions .....................................................  210
  10.2.5 Process Limit Count and IMS Quick Rescheduling ...............  211
  10.2.6 Parallel Scheduling .............................................................  212
10.3 Scheduling Failures .....................................................................  213
  10.3.1 External Conflicts ................................................................  213
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3.2 Internal Conflicts</td>
<td>213</td>
</tr>
<tr>
<td>10.4 Scheduling Completion</td>
<td>214</td>
</tr>
<tr>
<td>10.5 Region States</td>
<td>214</td>
</tr>
<tr>
<td>10.5.1 Region Idle</td>
<td>214</td>
</tr>
<tr>
<td>10.5.2 Region Occupied</td>
<td>215</td>
</tr>
<tr>
<td>10.5.3 IMS Monitor Reports</td>
<td>215</td>
</tr>
<tr>
<td>10.6 Monitoring Scheduling Failures</td>
<td>216</td>
</tr>
<tr>
<td>10.6.1 IMS Monitor</td>
<td>216</td>
</tr>
<tr>
<td>10.6.2 IMSPARS</td>
<td>218</td>
</tr>
<tr>
<td>10.6.3 Summary of Terminology</td>
<td>221</td>
</tr>
<tr>
<td>10.7 Schedule End to First DL/I Call</td>
<td>221</td>
</tr>
<tr>
<td>10.8 Program Load Options</td>
<td>221</td>
</tr>
<tr>
<td>10.8.1 Standard Program Fetch</td>
<td>222</td>
</tr>
<tr>
<td>10.8.2 Preload</td>
<td>224</td>
</tr>
<tr>
<td>10.8.3 Library Lookaside/Virtual Lookaside Facility</td>
<td>226</td>
</tr>
<tr>
<td>10.8.4 Virtual Fetch</td>
<td>229</td>
</tr>
<tr>
<td>10.8.5 Comparison of Program Loading Options</td>
<td>231</td>
</tr>
<tr>
<td>10.9 IMS Application Program Execution</td>
<td>233</td>
</tr>
<tr>
<td>10.9.1 Paging-in the Dependent Regions</td>
<td>233</td>
</tr>
<tr>
<td>10.9.2 Dependent Region Dispatching Priorities</td>
<td>234</td>
</tr>
<tr>
<td>10.9.3 COBOL Options</td>
<td>234</td>
</tr>
<tr>
<td>10.9.4 Common Execution Environment (LE/370)</td>
<td>235</td>
</tr>
<tr>
<td>10.9.5 Program Elapsed Time</td>
<td>236</td>
</tr>
<tr>
<td>10.9.6 Detailed Monitoring of Program Execution</td>
<td>236</td>
</tr>
<tr>
<td>Chapter 11. IMS Database Access Performance</td>
<td>237</td>
</tr>
<tr>
<td>11.1 Introduction to IMS Database Buffers</td>
<td>237</td>
</tr>
<tr>
<td>11.2 OSAM Databases</td>
<td>237</td>
</tr>
<tr>
<td>11.2.1 OSAM Buffer Pool</td>
<td>237</td>
</tr>
<tr>
<td>11.2.2 OSAM Sequential Buffering</td>
<td>238</td>
</tr>
<tr>
<td>11.3 VSAM Databases</td>
<td>239</td>
</tr>
<tr>
<td>11.3.1 VSAM Buffer Pool</td>
<td>240</td>
</tr>
<tr>
<td>11.3.2 VSAM Hiperspace Buffers</td>
<td>241</td>
</tr>
<tr>
<td>11.3.3 VSAM Background Write</td>
<td>243</td>
</tr>
<tr>
<td>11.4 IMS Fast Path Databases</td>
<td>245</td>
</tr>
<tr>
<td>11.4.1 Main Storage Database</td>
<td>245</td>
</tr>
<tr>
<td>11.4.2 Data Entry Database</td>
<td>245</td>
</tr>
<tr>
<td>11.4.3 DEDB Sequential Processing</td>
<td>247</td>
</tr>
<tr>
<td>11.4.4 IMS Fast Path Buffers</td>
<td>248</td>
</tr>
<tr>
<td>11.5 IMS Database Buffer Handlers</td>
<td>249</td>
</tr>
<tr>
<td>11.5.1 IMS Batch</td>
<td>249</td>
</tr>
<tr>
<td>11.5.2 IMS DB/TM and DBCTL</td>
<td>251</td>
</tr>
<tr>
<td>11.5.3 Buffer Pool Management</td>
<td>253</td>
</tr>
<tr>
<td>11.6 IMS Reports and Statistics</td>
<td>265</td>
</tr>
<tr>
<td>11.6.1 IMS Monitor VSAM Buffer Pool Report</td>
<td>265</td>
</tr>
<tr>
<td>11.6.2 IMS Monitor OSAM Buffer Pool Report</td>
<td>269</td>
</tr>
<tr>
<td>11.6.3 DB Monitor Program I/O Report (for Batch DL/I)</td>
<td>272</td>
</tr>
<tr>
<td>11.6.4 IMS Monitor DL/I Call Summary Report</td>
<td>273</td>
</tr>
<tr>
<td>11.6.5 Database Monitor VSAM Statistics</td>
<td>275</td>
</tr>
<tr>
<td>11.6.6 IMS /Display Pool Command</td>
<td>276</td>
</tr>
<tr>
<td>Chapter 12. IMS Application and Database Design for Performance</td>
<td>279</td>
</tr>
<tr>
<td>12.1 The Importance of Design Issues</td>
<td>279</td>
</tr>
<tr>
<td>12.2 Database Design</td>
<td>279</td>
</tr>
</tbody>
</table>
Chapter 14. IMS Database Control

14.1 CICS to DBCTL Interface and Interface Components .......................... 315
14.2 The DBCTL High Performance Environment ........................................ 316
  14.2.1 Unconstrained Virtual Storage .................................................. 316
  14.2.2 Parallel DL/I Environment ....................................................... 317
  14.2.3 Access to Data Entry Databases ............................................... 317
  14.2.4 Using BMPs for Parallel Batch Processing ................................... 318
  14.2.5 Potential for Reduced CPU Cost ............................................... 318
14.3 CICS—DBCTL Performance Monitoring ............................................. 319
  14.3.1 CICS/DBCTL Events ............................................................. 319
  14.3.2 IMS /DISPLAY Commands ...................................................... 319

14.4 Performance Parameters in CICS and IMS ......................................... 323
  14.4.1 DL/I Thread Allocation .......................................................... 324
  14.4.2 DRA MINTHRD and MAXTHRD Parameters .................................... 324
  14.4.3 IMS MAXREGN Parameter ....................................................... 325
  14.4.4 CICS MXT SIT Parameter ....................................................... 325
  14.4.5 Storage Requirements .......................................................... 325
  14.4.6 IMS Database Buffer Pools .................................................... 326

14.5 MVS Considerations ........................................................................... 327
  14.5.1 Dispatching Priority ............................................................... 327
  14.5.2 CPU Time .................................................................................. 327

Chapter 15. Performance of IMS with DB2 .............................................. 329

15.1 IMS External Subsystem Attach Facility ........................................... 329
  15.1.1 Subsystem Member ..................................................................... 330
15.2 Tuning the External Subsystem Attach Facility ................................... 331
  15.2.1 Thread Management ................................................................... 332
  15.2.2 DB2 Lock Management ............................................................. 333
  15.2.3 Static versus Dynamic SQL ....................................................... 335
  15.2.4 Security Controls ....................................................................... 336
  15.2.5 Dispatching Priority ................................................................... 336
15.3 Tools for Monitoring ........................................................................... 337
  15.3.1 IMS Monitor .............................................................................. 338
  15.3.2 DB2 Performance Monitor .......................................................... 344

Appendix A. Use of IMSPARS in Performance Analysis ......................... 347
  A.1 IMSPARS Reports ............................................................................ 347
  A.2 Transit Time Analysis Report .......................................................... 347
  A.3 Transaction Transit Log .................................................................... 347
  A.4 Region Histogram ........................................................................... 348
  A.5 CPU Usage Report ........................................................................... 352
  A.6 Internal Resource Usage Reports ...................................................... 354
    A.6.1 DMB Pool Statistics ................................................................... 354
    A.6.2 IMSPARS DASD Log Statistics .................................................. 354
    A.6.3 IMSPARS Program Isolation Enqueue/Dequeue Report .............. 354

Appendix B. Use of IMSASAP II .............................................................. 361
  B.1 IMS Resources ................................................................................ 361
    B.1.1 IMS System Transaction Profile ............................................... 361
    B.1.2 Region Performance Indicators .................................................. 371
    B.1.3 I/O IWAIT Times ....................................................................... 373
    B.1.4 IMS Pool Space ......................................................................... 373
  B.2 IMS Application/Database Resource ................................................ 373
    B.2.1 Application Programs ............................................................... 373
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.2.2 Databases</td>
<td>375</td>
</tr>
<tr>
<td>B.2.3 Program Trace Report</td>
<td>375</td>
</tr>
<tr>
<td><strong>Appendix C. Worksheets</strong></td>
<td>377</td>
</tr>
<tr>
<td>C.1 CPU Resource Worksheet</td>
<td>377</td>
</tr>
<tr>
<td>C.2 I/O Resource Worksheet</td>
<td>379</td>
</tr>
<tr>
<td>C.3 Real Storage Resource Worksheet</td>
<td>381</td>
</tr>
<tr>
<td>C.4 IMS Resource Worksheet</td>
<td>383</td>
</tr>
<tr>
<td>C.5 IMS Application Program and Database Worksheet</td>
<td>386</td>
</tr>
<tr>
<td><strong>Appendix D. Special Notices</strong></td>
<td>389</td>
</tr>
<tr>
<td><strong>Appendix E. Related Publications</strong></td>
<td>391</td>
</tr>
<tr>
<td>E.1 International Technical Support Organization Publications</td>
<td>391</td>
</tr>
<tr>
<td>E.2 Redbooks on CD-ROMs</td>
<td>391</td>
</tr>
<tr>
<td>E.3 Other Publications</td>
<td>391</td>
</tr>
<tr>
<td><strong>How to Get ITSO Redbooks</strong></td>
<td>393</td>
</tr>
<tr>
<td>How IBM Employees Can Get ITSO Redbooks</td>
<td>393</td>
</tr>
<tr>
<td>How Customers Can Get ITSO Redbooks</td>
<td>394</td>
</tr>
<tr>
<td>IBM Redbook Order Form</td>
<td>395</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td>397</td>
</tr>
<tr>
<td><strong>List of Abbreviations</strong></td>
<td>399</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>403</td>
</tr>
<tr>
<td><strong>ITSO Redbook Evaluation</strong></td>
<td>405</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Identifying the Bottleneck</td>
</tr>
<tr>
<td>2</td>
<td>CPU Seconds per Transaction</td>
</tr>
<tr>
<td>3</td>
<td>Estimated Response Time</td>
</tr>
<tr>
<td>4</td>
<td>Full Function Flow of an IMS Transaction</td>
</tr>
<tr>
<td>5</td>
<td>Fast Path Transaction Flow</td>
</tr>
<tr>
<td>6</td>
<td>Starting and Stopping the IMS Monitor</td>
</tr>
<tr>
<td>7</td>
<td>Printing the IMS Monitor Call Summary Report Using DFSUTR0</td>
</tr>
<tr>
<td>8</td>
<td>RMF Monitor I CPU Activity Report</td>
</tr>
<tr>
<td>9</td>
<td>RMF Monitor II SRCS Report</td>
</tr>
<tr>
<td>10</td>
<td>RMF Monitor II - ASD Compatibility Mode</td>
</tr>
<tr>
<td>11</td>
<td>RMF Monitor I Channel Path Activity Report</td>
</tr>
<tr>
<td>12</td>
<td>RMF Monitor I - Direct Access Storage Device Activity</td>
</tr>
<tr>
<td>13</td>
<td>Caching Subsystem Summary Report</td>
</tr>
<tr>
<td>14</td>
<td>RMF Monitor I Paging Report (Central Storage)</td>
</tr>
<tr>
<td>15</td>
<td>RMF Monitor I Paging Report (Expanded Storage)</td>
</tr>
<tr>
<td>16</td>
<td>RMF Monitor I Paging Report (Swap Placement)</td>
</tr>
<tr>
<td>17</td>
<td>RMF Monitor II SPAG Report</td>
</tr>
<tr>
<td>18</td>
<td>RMF Monitor I --- Work Load Activity Report (Part I By Groups)</td>
</tr>
<tr>
<td>19</td>
<td>RMF Monitor I --- Workload Activity Report (Part II By Policy)</td>
</tr>
<tr>
<td>20</td>
<td>RMF Monitor II ARD Report</td>
</tr>
<tr>
<td>21</td>
<td>IMS Monitor Run Profile Report</td>
</tr>
<tr>
<td>22</td>
<td>IMS Monitor Program Summary Report</td>
</tr>
<tr>
<td>23</td>
<td>IMS Monitor OSAM Buffer Pool Report</td>
</tr>
<tr>
<td>24</td>
<td>IMS Monitor VSAM Buffer Pool Report</td>
</tr>
<tr>
<td>25</td>
<td>IMS Monitor Region IWAIT Report</td>
</tr>
<tr>
<td>26</td>
<td>IMS Monitor Region Summary Report — Scheduling and Termination</td>
</tr>
<tr>
<td>27</td>
<td>IMS Monitor Region Summary Report — Schedule to First Call</td>
</tr>
<tr>
<td>28</td>
<td>IMS Monitor Region Summary Report — Elapsed Execution</td>
</tr>
<tr>
<td>29</td>
<td>IMS Monitor Region Summary Report — Idle for Intent and Region</td>
</tr>
<tr>
<td>30</td>
<td>IMS Monitor Programs by Region Report</td>
</tr>
<tr>
<td>31</td>
<td>IMS Command /DIS POOL ALL</td>
</tr>
<tr>
<td>32</td>
<td>Message Format Buffer Pool Report</td>
</tr>
<tr>
<td>33</td>
<td>IMS Monitor Communications IWAIT Report</td>
</tr>
<tr>
<td>34</td>
<td>IMS Monitor Communication Summary</td>
</tr>
<tr>
<td>35</td>
<td>IMS Monitor Line Functions</td>
</tr>
<tr>
<td>36</td>
<td>IMS Monitor DL/I Call Summary Report</td>
</tr>
<tr>
<td>37</td>
<td>IMS Elapsed Time</td>
</tr>
<tr>
<td>38</td>
<td>IMS Monitor Program I/O Report</td>
</tr>
<tr>
<td>39</td>
<td>IMS Key Performance Indicators</td>
</tr>
<tr>
<td>40</td>
<td>Formula for Determining the Number of Message Regions</td>
</tr>
<tr>
<td>41</td>
<td>CPU Performance Identification</td>
</tr>
<tr>
<td>42</td>
<td>Suggested IMS Dispatching Priorities</td>
</tr>
<tr>
<td>43</td>
<td>IMS Monitor Latch Conflict Report</td>
</tr>
<tr>
<td>44</td>
<td>I/O Performance Indicators</td>
</tr>
<tr>
<td>45</td>
<td>Identifying Channel-Busy Problems</td>
</tr>
<tr>
<td>46</td>
<td>Identifying Channel-Imbalance Remedies</td>
</tr>
<tr>
<td>47</td>
<td>Identifying Remedies to Shared DASD Problems</td>
</tr>
<tr>
<td>48</td>
<td>Addressing IMS Data Set Problems</td>
</tr>
<tr>
<td>49</td>
<td>Identifying Storage and Pool Problems</td>
</tr>
</tbody>
</table>
104. IMS/DB2 Dispatching Priorities Table ............................................. 337
105. IMS Monitor Region Summary DB2 Service and Command Calls ... 339
106. IMS Monitor Region Summary DB2 Normal Calls ......................... 340
107. IMS Monitor Region IWAIT (Showing DB2 IWAITs) ....................... 341
108. IMS Monitor Region Summary (Showing DB2 IWAITs) .................... 342
109. IMS Monitor Call Summary (Showing DB2 Calls) .......................... 343
110. IMSPARS Transit Time Analysis Report ........................................ 349
111. IMSPARS Transit Log Report ....................................................... 350
112. IMSPARS Region Histogram Report ............................................ 351
113. IMSPARS CPU Usage Report ....................................................... 353
114. IMSPARS DMB Pool Report and IMSPARS DASD Log Statistics Report 355
115. IMSPARS PI Enqueue/Dequeue and DL/I Call Reports — Program Isolation Statistics .............................................................. 356
116. IMSPARS PI Enqueue/Dequeue and DL/I Call Reports — DL/I Call Statistics ................................................................. 357
117. IMSPARS VSAM Buffer Pool Statistics Report ................................. 358
118. IMSPARS Message Queue Utilization Report ................................. 359
119. IMSASAP II Region Summary — Transaction ................................ 363
120. IMSASAP II Region Summary — Scheduling ................................ 364
121. IMSASAP II Region Summary — Call Information ......................... 365
122. IMSASAP II Region Summary — Region Distribution .................... 366
123. IMSASAP II Region Summary — Region Detail .............................. 367
124. IMSASAP II Total System IWAIT Summary ................................ 370
125. Number of I/Os Per Transaction ................................................... 371
126. IMSASAP II Program Summary .................................................... 372
127. IMSASAP II IWAIT Summary ........................................................ 374
128. CPU Resource Worksheet ............................................................ 378
129. I/O Resource Worksheet ............................................................. 380
130. Real Storage Resource Worksheet ............................................... 382
131. IMS Resource Worksheet (1 of 2) ............................................... 384
132. IMS Resource Worksheet (2 of 2) ............................................... 385
133. IMS Application Program and Database Worksheet ....................... 387
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>IMS PROCLIB Members</td>
<td>17</td>
</tr>
<tr>
<td>2.</td>
<td>MVS/ESA SYS1.PARMLIB Members Related to IMS Performance</td>
<td>18</td>
</tr>
<tr>
<td>3.</td>
<td>IMSASAP II Reports</td>
<td>19</td>
</tr>
<tr>
<td>4.</td>
<td>IMSPARS Reports</td>
<td>19</td>
</tr>
<tr>
<td>5.</td>
<td>RMF Monitor II Reports</td>
<td>20</td>
</tr>
<tr>
<td>6.</td>
<td>IMSPARS Reports Run Using IMS Logs</td>
<td>21</td>
</tr>
<tr>
<td>7.</td>
<td>MVS Data Collection Routines Summary</td>
<td>36</td>
</tr>
<tr>
<td>8.</td>
<td>IMS Data Collection Routines</td>
<td>62</td>
</tr>
<tr>
<td>9.</td>
<td>DASD Usage</td>
<td>89</td>
</tr>
<tr>
<td>10.</td>
<td>IMS Batch DL/I Swapping</td>
<td>141</td>
</tr>
<tr>
<td>11.</td>
<td>IMS Control Blocks</td>
<td>144</td>
</tr>
<tr>
<td>12.</td>
<td>List of Modules That Can Be Loaded into MLPA with IMS 5.1</td>
<td>157</td>
</tr>
<tr>
<td>13.</td>
<td>DFSPOOL Pool Locations</td>
<td>170</td>
</tr>
<tr>
<td>14.</td>
<td>Message Prefix Sizes in IMS/ESA</td>
<td>180</td>
</tr>
<tr>
<td>15.</td>
<td>IMS Application Program Loading Matrix</td>
<td>232</td>
</tr>
<tr>
<td>16.</td>
<td>Comparison of Program Load Times</td>
<td>233</td>
</tr>
<tr>
<td>17.</td>
<td>APPC/IMS Implicit Mode Recommendations</td>
<td>304</td>
</tr>
</tbody>
</table>
Preface

This redbook provides IMS/ESA system programmers, database administrators and application designers with guidelines for evaluating IMS/ESA system performance and tuning the IMS environment. Subsystems such as CICS/ESA and the DB2/ESA interface with IMS/ESA are also discussed, as are a variety of performance topics.

This redbook should also help the IBM Representative to understand how IMS/ESA functions and be able to help customers with various performance topics.

An in-depth knowledge of IMS/ESA is assumed.

How This Redbook Is Organized

This redbook contains 410 pages. It is organized as follows:

- Chapter 1, “Performance Monitoring and Tuning Overview”
  This chapter describes IMS/ESA system performance monitoring and tuning methodology and information sources.
- Chapter 2, “IMS System Parameters and Implementation Options”
  This chapter summarizes IMS/ESA system parameters related to performance.
- Chapter 3, “Monitoring the IMS Environment”
  This chapter describes the IMS monitoring environment.
- Chapter 4, “Monitoring the IMS Subsystem”
  This chapter presents the IMS monitoring tools.
- Chapter 5, “Identifying IMS Performance Problems”
  This chapter describes methods of identifying performance problems.
- Chapter 6, “Understanding CPU Resource Problems”
  This chapter describes MVS CPU resource related to IMS/ESA.
- Chapter 7, “Understanding Problems with the I/O Subsystem”
  This chapter describes the MVS input/output subsystem related to IMS/ESA.
- Chapter 8, “Understanding IMS Use of Real Storage and Pools”
  This chapter describes MVS related to IMS/ESA storage structure and performance.
- Chapter 9, “Understanding IMS Buffer Pool Storage”
  This chapter describes IMS/ESA internal storage and pools structure and performance.
- Chapter 10, “Scheduling and Dependent Region Considerations”
  This chapter describes IMS/ESA scheduling.
- Chapter 11, “IMS Database Access Performance”
  This chapter describes topics related to IMS/ESA database access.
- Chapter 12, “IMS Application and Database Design for Performance”
This chapter discuss application design.

- Chapter 13, “Database Recovery Control (DBRC) Performance”
  This chapter discuss DBRC performance topics.
- Chapter 14, “IMS Database Control”
  This chapter describes IMS/ESA DBCTL performance.
- Chapter 15, “Performance of IMS with DB2”
  This chapter describes the IMS/ESA interface with DB2/ESA.
- Appendix A, “Use of IMSPARS in Performance Analysis”
  This appendix presents sample output of IMSPARS.
- Appendix B, “Use of IMSASAP II”
  This appendix presents sample output of IMSASAP II.
- Appendix C, “Worksheets”
  This appendix presents sample worksheets for reference.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose Center.

Scott Chen is a member of Data Management of ITSO San Jose center. He has worked with IMS, both in application development and system programming, for over 15 years. He also works extensively in the area of MVS, DB2, CICS, and documentation image management.

Alan Cooper is an IMS Specialist Systems Engineer in the United Kingdom. He has worked with IMS for over 20 years, and is currently in the Installation Support Center, Hursley, where he is a member of the IMS support team providing technical support and education to IMS customers throughout Europe. Within IMS, Alan has taken a special interest in Fast Path and Performance. He has written many articles and presentations, and in particular was the principal contributor to the IMS Version 5 Guide.

Bob Gendry is a Senior OSS/Software Analyst with the IMS Consulting and Services Department at the Dallas Systems Center. His was first introduced to IMS as an IBM Systems Engineer in 1972 and has been working with IMS ever since. In 1978 he joined the IMS technical support group at the Palo Alto Systems Center, which moved to Dallas, Texas in 1980. While at the Systems Center, Bob's interests and assignments have been many and varied. He has developed a practical methodology for IMS performance analysis, assisted in the analysis and tuning of many users' IMS systems, and made numerous presentations on IMS performance at GUIDE, SHARE, the IMS Technical Conference, and other IMS user groups.

Hélène Lyon is an Advisory Systems Engineer with IBM France. She is currently the IMS Product Manager for IBM France, selling, marketing and supporting pre-sale on the IMS family of products and the IMS Tools from Icing Software. She has written several redbooks, and was a significant contributor to the IMS Version 5 Guide.
Geoff Nicholls is an Advisory Systems Engineer at the International Technical Support Organization, San Jose Center. Geoff has a Science degree from the University of Melbourne, Australia, majoring in Computer Science. He worked as an application programmer and database administrator for several Insurance companies before specializing in database and transaction management systems with Unisys and IBM. Since joining IBM in 1989, Geoff has worked extensively with IMS customers in Australia and throughout Asia, mostly with Fast Path.

Jean-Pierre De Villiers was an Advisory Technical Specialist in IBM South Africa. He has 8 years of experience in the IMS field. He holds a degree in Economics from Rand Afrikaans University. His areas of expertise include IMS systems programming. He has participated in writing several IMS redbooks, including MSC in a Sysplex Environment, and IMS Sysplex Data Sharing.

The advisor for the second edition of this redbook was

Alan Tippett, ITSO San Jose center.

Thanks to the following people for their invaluable contributions to this project:

Geoff Nicholls
IMS Specialist; The Application Development and Data Management
ITSO Center, San Jose

George Andreadis
Harley Beier
Ron Blake
Attila J. Fogarasi
Janna Mansker
Karen Ranson
Frank Ricchio
David Voss
Steve Williamson
IBM Santa Teresa Laboratory

Jim Boyle
IBM Australia
Peter Sadler
IBM UK
Nobuyuki Shimizu
IBM Japan

Robert Gendry
Bill Keene
Dallas Systems Center

ABSA Bank, South Africa
ANZ Bank, Australia
Barclays Bank, UK

Thanks to Maggie Cutler and Shirley Weinland Hentzell for editing the document.
Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 405 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:
  - For Internet users: http://www.redbooks.ibm.com
  - For IBM Intranet users: http://w3.itso.ibm.com
- Send us a note at the following address:
  - redbook@vnet.ibm.com
The performance of an IMS system is a very subjective thing that depends on a great number of variables. The variables may be found in the IMS system itself, the application programs, the database design, the operating system, the hardware configuration, or any combination of these areas. These variables include the system resources available and the user's options in the allocation of the resources to the IMS transaction workload. Options can be selected intelligently only if the user has established performance objectives and knows how the current system is performing, and why it is performing that way. Software monitoring is the only good means of understanding the current system.

This chapter gives some insight into establishing performance objectives and explains the three basic reasons for monitoring IMS systems:

- To check whether performance objectives are being met
- To tune the system if performance problems exist, or after a change in the system workload
- To establish base profiles for tuning comparisons and capacity planning

Each of these is explained in detail in this chapter.

IMS performance monitoring is the measurement of transaction profiles and the quantification of resource limitations on transaction time-in-system. Time-in-system is defined as the time the transaction is enqueued on the IMS input queue until the time the output message is sent back to the inputting terminal, to another terminal, or to a program. IMS tuning is the allocation of limited resources, by trading real or virtual storage for I/O, for expedient transaction processing.

1.1 Performance Objectives

Performance objectives must be defined by your installation prior to implementing performance monitoring procedures. IMS performance objectives might be stated in a variety of ways:

1. "Good response time."
   
   What is good?

2. "Response time of 3 seconds or less."
   
   What is response time? How is it measured?

3. "Messages per day, hour, minute, or second."
   
   What is a message? How many characters?

4. "Transactions per second."
   
   What is a transaction? How many messages? How much work does it generate?

5. "Two transactions per second with an average response time of three seconds."

Are three seconds possible for these transactions?
Before it is possible to define performance objectives, it is necessary to understand about transactions and transaction profiles.

### 1.2 Transaction Profiles

The unit of work by which most IMS systems are measured is the transaction. An IMS transaction is a message from either a terminal or an application program that causes application program logic to be executed. IMS performance is stated in terms of transactions rather than messages or screens.

#### 1.2.1 Transaction Variability

The definition of an IMS transaction must be further quantified before it can be used as a performance objective criterion. In many cases, the resource requirements of a transaction can vary significantly from one execution to another, as illustrated by the following examples.

1.2.1.1 Transaction Design

A transaction for an account balance inquiry by a bank teller may require one database call and, if the account number is known, one database I/O. However, the transaction may be coded to handle the case when the account number is unknown, in which it may take several calls and many I/Os to access the balance.

An order-entry application may be designed to enter one or to enter multiple line items per transaction. The entry with multiple line items causes more work to be done in one transaction than the entry with only one line item, but it may be more efficient overall for a business transaction.

1.2.1.2 Variation in Data Accessibility

Even when the application logic is identical, the same DL/I call can vary greatly in the amount of resource it consumes. Performance depends on the database record size, the length of twin or synonym chains, whether data is in the most desirable block, whether the target data exists, and other factors.

1.2.1.3 Variation over Time

Variation over time can have several causes:

- An application may have a different transaction mix at different times of the day, week, month, or year.
- New applications with a lower or higher profile may be added to the system.
- A database may become disorganized, causing the I/O profile to increase.

#### 1.2.2 Types of Profiles

The amount of work a transaction does can be quantified by the number of database calls issued and the number of I/Os generated to service those calls. This is referred to as a transaction profile. Some transaction profiles are exactly the same for every execution of the transaction. However, most transaction profiles vary for each execution depending on the input data, the location of the database segments, or various other application design factors.

An application transaction profile is the average transaction profile for all transaction types executed in the application. Following this logic one step further, the system
transaction profile for the IMS system may be defined as the average number of
database calls and I/Os for all transactions executed on the system in a given time
interval. IMS class scheduling causes message processing regions to execute
different transaction mixes. Therefore, different message regions may have
different transaction profiles. IMS system transaction profiles will usually change
when the workload changes.

The use of monitoring to establish transaction profiles is discussed later in this
chapter, and the actual measurement of an IMS system profile is explained in 4.2.2,
“IMS System Transaction Profile” on page 69.

1.3 Setting Performance Objectives

Once a transaction has been defined as a quantitative amount of work to be done,
a performance objective can be stated as numbers of transactions per time interval,
together with the time required to execute each transaction. So IMS performance
objectives are stated in terms of transactions per second, qualified by a transaction
profile, with response time requirements.

More specifically, it is recommended that performance objectives for an IMS
message processing system:

- Be defined in terms of both average and 90 percentile response time
  requirements, based on a specified profile.
- Differentiate between internal response time (while the message is within the
  MVS system) and end-user response time (which includes the network
  transmission times).
- Recognize when the business transaction (as perceived by the end user)
  requires several IMS transactions to be executed.
- Be achievable with specified throughput levels.

There is no room for ambiguity. Consider the following: an IMS performance
objective is stated as 3 transactions per second, with a transaction profile of 30
database calls and 15 I/Os and an average response time of less than 3 seconds.

This response time must be defined as either queue-to-queue or end-user terminal
response time. It is important to be aware of the expected variation in the
transaction profile. Finally, the following question needs to be asked: “Is this
response time reasonable?”

1.3.1 Resource Requirements

The response time of an IMS transaction is determined by the transaction's
time-in-system. The time-in-system for a transaction is a function of the work to be
done and the resources available to do that work. The resources required by an
IMS transaction include:

- CPU cycles to process the transaction, associated I/O activity, and support
  functions.
- Real storage to hold the processing code, tables, control blocks, I/O buffers.
- I/O configuration for the external storage of data and the network for
  transmission to and from the source terminal.
Virtual storage to increase the size of the buffer pools or to increase the number of partition specification tables (PSTs) to support the workload.

The lack of availability of one of these resources, when it is required by a transaction, increases the transaction's time-in-system. IMS is designed to process each database call and the I/Os to service that call serially for each transaction, so the time-in-system for each transaction is directly proportional to the amount of work it does.

1.4 Reasons for Monitoring IMS

Four reasons for monitoring the IMS system are suggested.

1.4.1 Monitoring to Check Achievement of Performance Objectives

Many IMS systems get monitored only when end users complain that response time has degraded or become erratic. The following questions are typically asked:

- How bad is it (relatively)?
- How long has it been bad?
- Has the system workload changed recently?
- Does the database need to be reorganized?
- Have there been recent hardware or software changes?

If the system is not monitored on a regular basis, it may be impossible to answer these questions. More important, many problems can be predicted and avoided when the IMS system is regularly monitored.

1.4.1.1 Regular Monitoring

Frequent monitoring of the system and plotting of key performance indicators show changes or trends that affect response time before end users notice. The frequency of monitoring depends on the volatility of the system. Run the IMS Monitor once or twice a day at both peak and nonpeak periods to establish a base set of indicators. Once the workload pattern of the system is known, you can reduce the frequency of monitoring to once or twice a week. It may be necessary to increase the frequency as the workload changes. Monitoring the system and plotting the key performance indicators take very little time, once the procedures have been established and automated.

Monitor both peak and nonpeak periods because the use of resources changes for each of these periods. The peak period requires higher region occupancy and additional DASD I/O, whereas the nonpeak period tends to page more and require less DASD I/O.

Note any gradual or significant changes in the performance indicators, and make an effort to explain the cause of each change. Once the cause is identified, tuning can compensate for the change. Changes in IMS performance can result from non-IMS workload factors as well. The IMS Monitor can pinpoint operating system and hardware resource limitations. Chapter 3, “Monitoring the IMS Environment” on page 35 provides guideline numbers for the key performance indicators and explains how to identify various resource limitations, and why those limitations affect IMS performance.
Log data is used to track the performance of an IMS system. IMSPARs, or any other analysis program that analyzes the IMS log, can be used to report performance indicators for IMS transactions. These indicators, of which there can be many different types, are used to track IMS system performance. Users of IMS Fast Path facilities can also analyze the log using the Fast Path Log Analysis utility (DBFULTA0).

If indicators of changed performance are present, you can monitor the system with more detailed monitors, such as the IMS Monitor and generalized trace facility (GTF). This allows you to find and correct performance problems before the end user notices an increase in response time.

1.4.2 Monitoring for Tuning

When IMS performance is poor, for whatever reason, it becomes necessary to monitor the system to identify the resource bottlenecks that are causing the poor performance. Examples of system changes that make tuning necessary include these:

- A change in the hardware configuration
- A new release of MVS
- A new release of IMS
- A new IMS application
- An increase in the paging rate caused by an increase in TSO, batch, or other subsystem workload
- Allocation of new data sets
- Changes in the IMS database or application programs
- An unanticipated increase in the IMS workload

Because many performance problems are caused by factors outside of IMS, the MVS environment must be monitored as well.

The IMS Monitor records IMS events. The duration of these events can be used to infer lack of CPU cycles, high paging activity, and DASD I/O interference. Other software monitoring tools must be used to specifically identify these external bottlenecks. Refer to Table 7 on page 36 and Table 8 on page 62 for tools that can be used in performance monitoring.

Each software monitor uses system resources; therefore, it is not practical to run all of them at once. To tune IMS, it is desirable to have IMS Monitor and resource management facility (RMF) reports that cover approximately the same time period when performance problems are occurring.

Tuning the IMS system is accomplished by removing the bottlenecks that cause the transaction time-in-system to be longer than necessary. The most significant bottleneck must be identified and removed to improve performance. Changing variables that affect lesser bottlenecks does not have a significant impact on performance (see Figure 1 on page 6).
Example:

In attempting to get fans in and out of a ball game faster, it does not help to widen the walk ramps if the real bottleneck is that there are only two traffic lanes to the parking lot. The traffic jam is the significant performance indicator.

Figure 1. Identifying the Bottleneck

A tuned IMS system is one that gets maximum transaction throughput with minimum or acceptable response time with the hardware and software resources available.

1.4.3 Monitoring to Establish Base Profiles

A base transaction profile is the transaction profile in a stand-alone environment, without interference. It is desirable to build base profiles for those transactions that have the highest volume and/or generate the most work in terms of calls and I/Os.

The base profiles for these transactions can be used for two purposes:

- To examine the transaction's workload to see if there is any way to reduce it. This type of application change has the greatest impact on reducing the system transaction profile.
- To compare a base transaction profile with the same transaction profile in a production environment. This procedure can be used to locate and quantify differences caused by resource contention.

You can obtain a base transaction profile by bringing up IMS on a dedicated machine (one choice might be third shift or Saturday night, with no batch and no shared DASD). Execute at least one of each type of transaction to establish the environment (open databases, initialize BLDL list, load IMS in main storage); then start the IMS Monitor and execute each transaction to be measured. Enter one transaction at a time, waiting for a response before entering the next.

You can obtain a base system profile by executing the production IMS system in a stand-alone environment (that is, where no other work is contending with the IMS service). This is usually not operationally possible for extended periods of time, but with a little planning, most installations can afford to stop all other work for 10 or 15 minutes. A base system profile shows what the IMS system is capable of without external resource contention from other tasks or jobs in the system. The profile also can be compared with a true production-system profile to identify and quantify resource contention.
1.5 Capacity Planning

System transaction profiles can be used for capacity planning. When measurements taken and validated indicate that the increase in CPU use is nearly linear with the increase in transactions per second, then, if the additional transactions have the same profile, the remedy is likely to be a more powerful CPU.

For any given processor, the limiting hardware resource is processing power (CPU cycles). Therefore, if the transaction profile and CPU utilization are known for a particular transaction arrival rate, it is relatively simple to predict the capacity of the system for a transaction volume increase of the same profile. The impact of adding a new application to the system can also be predicted if the application transaction profile and volume can be accurately estimated.

These predictions assume dedicated IMS systems. They also assume adequate real storage, virtual storage, and I/O configuration to handle the increase. In making capacity predictions, carefully consider the remaining available I/O and main storage resources, because contention for these resources has a severe impact on performance no matter how much spare CPU capacity is available.

Some capacity planning can also be done for new IMS systems when the application design has progressed to the point where transaction volumes, message lengths, and transaction profiles can be realistically predicted. The anticipated workload can be compared with a similar production system. An alternative approach is to use average weighted value estimates for significant IMS transaction activity, namely DL/I calls and I/Os. Figure 2 on page 9 represents average CPU costs for issuing DL/I calls and driving I/Os for an IMS Transaction Manager (TM) system, including all facilities. CPU time can be estimated by multiplying the unit values by the transaction profile counts. The example is theoretical only; it relates to a particular IBM processor and is not meant to be used for your capacity planning projections.

The base cost in time in Figure 2 on page 9 represents the time it takes to schedule the transaction, format the input/output message, perform logging, take the required checkpoints, and process the transactions. The time shown is not to be considered the exact amount required by IMS; it is simply an example of an estimate.

Unit values for functions vary with different IMS/ESA and MVS releases, so it is important to understand that this technique is to be used only for determining approximate capacity. Minimum anticipated response time also can be estimated by adding line transmission time for average message length, average DASD I/O time, and program load time to the CPU time.
The I/O elapsed times in Figure 3 are for an IBM 3390 DASD device.

<table>
<thead>
<tr>
<th>Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200 characters @ 4800 BAUD</td>
<td>2000 ms</td>
</tr>
<tr>
<td>15 I/Os @ 19 ms each</td>
<td>285 ms</td>
</tr>
<tr>
<td>CPU time from 1st Example</td>
<td>26 ms</td>
</tr>
<tr>
<td>Program load time</td>
<td>200 ms</td>
</tr>
<tr>
<td>Application program time</td>
<td>30 ms</td>
</tr>
<tr>
<td>MVS queueing and overhead</td>
<td>98 ms</td>
</tr>
<tr>
<td>Estimated Response Time</td>
<td>2639 ms or 2.64 seconds</td>
</tr>
</tbody>
</table>

Figure 3. Estimated Response Time

There is no guarantee that a production system will perform to these estimates. However, the technique is valid for estimating the CPU capacity and response time of a projected IMS application.

1.6 IMS Performance Management Overview

The ideal is to tune the IMS system to handle peak transaction arrival rates while achieving performance objectives. A system that is tuned to the peak arrival rate normally also provides the required performance service levels at nonpeak volumes. Exceptions to this are the introduction of multiple batch message processing (BMP) regions, changes in the non-IMS workload, or a significant decrease in arrival rate, which causes a higher IMS paging rate. To tune to the peak workload, measurements must be made at peak times to obtain a representative transaction profile.

Tuning IMS is a compromise between minimizing contention for system resources by MVS and other subsystems, and reducing the IMS transaction resource consumption:

- The contention for CPU cycles is addressed by the dispatching priorities of the IMS system address spaces: (control region, DL/I separate address space (DLISAS), and database recovery control (DBRC) and IMS dependent regions such as IMS Fast Path (IFP), message processing program (MPP), and batch message processing (BMP) relative to other subsystem components and operating system services. The use of CPU cycles by MVS can be addressed by minimizing the contention for main storage and for the I/O configuration.
- Main storage contention can be reduced by running fewer jobs, page-fixing less storage, and trimming working sets. In some cases, the solution is to add more real storage.
- I/O contention can be reduced by executing fewer I/Os, reconfiguring the DASD subsystem, using DASD Fast Write (DFW) and cache facilities, changing data set placement, and changing data control block (DCB) options. If demand paging is a bottleneck for the IMS system, additional or faster paging devices can help avoid performance problems for IMS transactions.

---

1 When using the MVS V5 Workload Manager, the dispatching priorities of the IMS address spaces are managed dynamically and transparently by MVS.
The IMS transaction profile can be improved by changing the application program design or database structure so that fewer DL/I calls and fewer database I/Os per call are issued.

- The number of I/Os per transaction can be reduced by adjusting IMS pool sizes and by making commonly used modules and data resident in virtual storage. Reducing in the number of I/Os can result in less CPU use as well as less I/O (channel, control unit, and string) contention.

### 1.7 IMS Events That Are Monitored

The Fast-Path component of IMS offers extremely high performance handling of databases and messages. This enhanced performance is achieved partly by implementing fewer options and reduced functions. In contrast, the non-Fast-Path part of IMS is referred to as *full function*. Data entry databases (DEDBs) and main storage databases (MSDBs) are Fast Path databases, whereas hierarchical direct access method (HDAM), hierarchical indexed direct access method (HiDAM), and hierarchical indexed sequential access method (HiSAM) are full-function databases. A full-function transaction uses the IMS message queues and is scheduled by the transaction scheduler into a message processing region (MPP, or message-driven BMP). The Fast Path message processing component is called the expedited message handler (EMH). Full-function transactions and Fast-Path transactions can use both full-function and Fast-Path databases, as appropriate.

The events that can be monitored differ significantly between full function and Fast Path. This is partly caused by architectural differences between the two types of processing, but it is primarily the result of the fact that the full-function process can write monitoring data to both the IMS log and the IMS Monitor. The IMS Monitor does not record any information about either Fast Path message processing (EMH) or Fast Path databases. Therefore, all Fast Path monitoring information must be taken from the IMS log.
1.7.1 IMS Full-Function Transactions

Each transaction goes through a distinct set of events as it is processed by the IMS system. The duration of these events is measured by the IMS Monitor and reported by the IMS Monitor Report Print utility (DFSUTR20). Input and output message handling, message format service (MFS), and message queueing are asynchronous to transaction processing. In this case, asynchronous means that the processing and I/O activities for these events are not associated with a message processing region (dependent region PST) and can proceed concurrently with the processing of other transactions. The events from application scheduling to termination are associated with a dependent region PST and are referred to as synchronous events.

Figure 4 on page 11 represents each monitored event for IMS full function, the major activity in each event, and the data sets and pools that are involved. The duration of each event is governed by the amount of processing, the number of I/Os, and the amount of interference caused by resource contention. CPU contention is reflected in IWAIT and NOT-IWAIT time, I/O contention results in longer I/O IWAIT times, and main storage contention causes an increase in paging time. This book is organized to separately address contention for the three critical machine resources: CPU cycles, I/O configuration, and real storage.

The following is a description of the events illustrated in Figure 4 on page 11 that constitute an IMS transaction:

1.7.1.1 Message Input

The IMS control of a transaction begins when the message is placed by VTAM in an IMS receive-any (RECANY) buffer. The message is moved to a buffer acquired in the High I/O Pool (HIOP) where it is edited by MFS routines, IMS basic edit or intersystem communication (ISC) edit (depending on terminal type and bypass MFS option, as appropriate). Once the edited message is in a format ready for processing, it is allocated a position on one of the message queue data sets. The message is moved to the message queue pool and enqueued on the scheduler message block (SMB). All of this activity contributes to end-user response time.

Usually this response time is short, but it can be significant if I/Os to the message queue data sets increase because not enough message queue buffers are defined to hold all queued input and output messages.

1.7.1.2 Scheduling

Scheduling can play an important role in the time-in-system of a transaction. If the transactions are defined with different classes and the message regions are defined to handle only specified classes, it is possible for some of the regions to be idle even though there are transactions on the input queue. The amount of time transactions spend on the input queue needs to be monitored, and evaluated if it becomes significant.

Other factors that contribute to transactions remaining on the message queue are:

- Scheduling I/Os to ACBLIB to load intent lists, program specification blocks (PSBs) and data management blocks (DMBs). DMBs are to be loaded only the

---

2 A long multisegment message may be allocated several records on several queue data sets.

3 The scheduler message block is the main control block for a transaction code and contains the anchor point for an input queue.
### Figure 4. Full Function Flow of an IMS Transaction

<table>
<thead>
<tr>
<th>EVENT</th>
<th>ACTIVITY</th>
<th>POOLS/ LISTS</th>
<th>DATA SETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Input</td>
<td>TP</td>
<td>RECANY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>HIOP</td>
<td>(VTAM)</td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>CIOP</td>
<td>LINES (BTAM)</td>
</tr>
<tr>
<td></td>
<td>MFS</td>
<td>MFP</td>
<td>FORMAT</td>
</tr>
<tr>
<td></td>
<td>SPA Get</td>
<td>SPAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queueing</td>
<td>QBUF</td>
<td>MSGQs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG</td>
<td>IMSLOG</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Scheduling</td>
<td>QBUF</td>
<td>MSGQs</td>
</tr>
<tr>
<td></td>
<td>PSB Load</td>
<td>PSBP</td>
<td>ACBLIB</td>
</tr>
<tr>
<td></td>
<td>DMB Load</td>
<td>DMBP</td>
<td>ACBLIB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG</td>
<td>IMSLOG</td>
</tr>
<tr>
<td>Schedule End to first DL/I Call</td>
<td>Program Load</td>
<td>BLDL</td>
<td>STEPLIBs</td>
</tr>
<tr>
<td></td>
<td>Initialization</td>
<td>VLF/LLA</td>
<td>Preload</td>
</tr>
<tr>
<td>Program Elapsed</td>
<td>DC Calls</td>
<td>QBUF</td>
<td>MSGQs</td>
</tr>
<tr>
<td></td>
<td>DL/I Elapsed</td>
<td>DB Calls</td>
<td>OSAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VSAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LOG</td>
</tr>
<tr>
<td></td>
<td>IWAIT Elapsed</td>
<td>SPA Insert</td>
<td>SPAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LOG</td>
</tr>
<tr>
<td></td>
<td>Sync Point</td>
<td>OSAM</td>
<td>Databases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VSAM</td>
<td>Databases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QBUF</td>
<td>MSGQs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG</td>
<td>IMSLOG</td>
</tr>
<tr>
<td>Termination</td>
<td>Message Out</td>
<td>Dequeueing</td>
<td>QBUF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MFS</td>
<td>MFP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP</td>
<td>CIOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TP</td>
<td>HIOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG</td>
<td>IMSLOG</td>
</tr>
</tbody>
</table>

First time a transaction is executed, or not at all for resident DMBs. Also, no I/O for the intent lists is expected if they are resident. For most IMS systems, it is not practical to have a PSB pool large enough for all PSBs to be resident; therefore, ACBLIB I/Os to load PSBs are to be expected for some transactions.
- Pool space failures. The PSB, PSB work pools (PSBW), and DMB pools must continually be monitored for space failures and increased in size if any failures occur.

- A transaction arrival rate such that a transaction arrives while another of the same type is executing, and the transaction cannot be scheduled in parallel or, if it can, no other region is available to process that transaction in parallel.

1.7.1.3 Schedule End to First DL/I Call
The events processed for schedule-end-to-first-DL/I call are the loading of the application program (if not preloaded) and the loading of any subroutines required before the first DL/I call to retrieve a message. Also included is the time required to initialize the program and open any non-DL/I files. Message-processing programs should not use non-DL/I files because of the heavy overhead of opening and closing them. Another factor that can influence this time is whether the application program issues any database DL/I calls before the first message-queue DL/I call. If this is done, the time to process this call is also included in the schedule-end-to-first-DL/I call time.

1.7.1.4 Program Elapsed Time
The program elapsed time can be divided into the following categories:

- Program (that is, non-DL/I) execution time for the application program to process its work, including the loading of subroutines.

- DL/I elapsed time, which is the time required to complete the DL/I calls. It has two components:
  - IWAIT time, which is the time DL/I has to wait for the I/O to complete or the time DL/I must wait to acquire a database record that another application program has enqueued (program isolation wait). The IWAIT elapsed time is typically the most significant component of transaction time.
  - NOT-IWAIT time, which is the DL/I code execution time and includes the time to search database buffers for the required blocks or control intervals (CIs). If a buffer pool is full of updated blocks or CIs, and the application requires a new block or CI, then a write is forced to make space in the pool. In this case, the DL/I NOT-IWAIT time includes the log write-ahead I/O time.

- Any of the above processing times (IMS or application) can involve waits (unseen by IMS) as a result of:
  - Waiting for a page fault to be resolved. This can be considerable if the paging subsystem is not adequately defined.
  - The processor executing in an overcommitted environment because MVS dispatching is not set up correctly.

Most message-driven programs are written to process multiple messages for each program schedule. After they finish processing one message, they reissue the IOPCB get-unique (GU) call. This drives synchronization point (sync point) processing.

**Full Function Sync Point Processing:** For typical transaction processing programs, this is when all the database update I/Os actually take place. IMS uses the standard two-phase commit in the following sequence:
1. Physically log all the database updates. Write out all buffers updated by the transaction.

2. If all resource managers have reported that Phase 1 was successful, then release locks on the updated data. If any resource manager reported that Phase 1 was unsuccessful, then back out the updates and release the locks.

Theoretically, it is not necessary to write out database updates at sync point time. DB2 for example, logs the updates at sync point to make them recoverable, but writes to the databases asynchronously. The DL/I strategy has the benefit that system emergency restart is simpler and quicker.

### 1.7.1.5 Termination

If the application does not force an explicit sync point with a message GU call, then the termination time includes the logging of the database updates and the writing of these updates to the database data sets.

More generally, termination is the short time required to tidy up the dependent region in order to prepare for the next program schedule. The output messages are then ready to be sent to their final destination.

### 1.7.2 IMS Fast Path Transactions

The flow of a Fast Path transaction through the IMS Transaction Manager is rather different from that of a full-function transaction. Also, as explained previously, all monitoring data is contained in the standard Fast Path log records.

Figure 5 on page 14 depicts the various facets of Fast Path, namely expedited message handling, data entry databases and main storage databases. With the IMS Fast Path EMH, a Fast Path transaction has the following processing stages:

1. The initial processing is exactly as for full function processing. The message arrives from VTAM in a RECANY buffer and is edited in the HIOP by MFS, basic edit, or ISC edit.

2. If necessary, a buffer is allocated in the EMH buffer pool, and the message is moved into it.

3. The user-written Fast Path Input Edit/Routing exit (DBFHAGU0) is called. If the transaction has been defined (in IMS stage 1) as a Fast-Path-potential transaction, then the exit can choose to have the message processed by full function IMS, in which case it is subsequently moved to a buffer in the message queue pool. Otherwise, the exit must determine the appropriate application program by specifying a routing code.

4. If the message is chosen to be processed by Fast Path or is Fast Path exclusive, IMS queues the message (in the EMH buffer) off the balancing group (BALG) associated with the specified routing code. The balancing group functions as a queue anchor point and can have several associated routing codes. One BALG exists for each IMS Fast Path program, and there multiple IFP regions may be running the same program, all servicing the same BALG when parallel scheduling is enabled. There is no priority scheduling scheme; waiting messages are processed in the order they arrive, often referred to as first-in, first-out (FIFO) scheduling.

---

4 MSDB has, for the most part, been superseded by the virtual storage option (VSO) for DEDB.
No log records are built at this time. However, at sync point, a log record is built that reflects this queueing of the input message. Consequently, from a performance monitoring point of view, this is the start of the in-queue time period.

5. When the next IFP region associated with the BLAG is ready, the transaction is processed. Note that Fast Path dependent regions always execute in wait-for-input (WFI) mode.

The time of the message GU is included in a log record (at sync point) and, from a performance monitoring perspective, represents the end of the in-queue time period and the start of the processing time period.

6. The transaction reply (IOPCB ISRT) is put into the EMH buffer allocated to the terminal.
7. The transaction is completed when the program reaches sync point. Again, the
time is logged, as it represents the end of the processing period and the start
of the out-queue time period. The events of Fast Path sync point are
considered in 1.7.2.1, “IMS Fast Path Sync Point and Associated Processing”
on page 15.

At this point, the program is ready to process the next message.
Asynchronously, the following events take place in parallel with the processing
of subsequent transactions.

8. The physical writes of the transaction's log records (built at sync point time) are
processed. This triggers the updating of the DEDBs by the output threads
(OTHREADs), and also causes the transaction's reply message (in the EMHB)
to be sent. For full function, the reply goes through the HIOP, where MFS
editing takes place if appropriate.

The time of this send is eventually logged in the dequeue log record, and
represents the end of the out-queue time period and the start of the output time
period.

9. The terminal acknowledges the reply (with an SNA response DR2, or optionally
for SLUTYPEP, with the next input message or an SNA RTR). IMS dequeues
the reply and builds a log record to this effect. From a performance monitoring
viewpoint, this represents the end of the output time period, and also the end of
the transaction.

Unlike full function (MPP) transactions, IFP transactions are scheduled without
using either the message queues or the priority scheduling algorithms, thus
avoiding potential I/O activity to the message queue data sets. The application
programs are always in wait-for-input mode, so the overhead of deleting and
loading programs, as well as managing and loading PSBs is avoided.

All updates to MSDBs and DEDBs are kept in buffers until the application program
reaches sync point. Unlike with full function, a Fast Path transaction's physical
database updates can never occur while that transaction is being processed.

1.7.2.1 IMS Fast Path Sync Point and Associated Processing
Fast Path sync point processing is also significantly different from full function sync
point processing. A key element of Fast Path sync point processing is that no I/Os
are required.

Fast Path Sync Point Processing: The Fast Path sync point is defined as the
next GU call for a message-driven program, or a SYNC or CHKP call for a BMP
using Fast Path facilities. Again, the sync point process has two phases:

1. The DEDB areas and MSDBs used by the application program are checked for
availability. Resource contentions, deadlocks, out-of-space conditions, and
MSDB field call verify failures can be discovered at this time and cause sync
point processing to be aborted. Locks on any unmodified CIs are released,
and the associated buffers are made available. Space required to store the
sequential dependents (SDEPs) in the current SDEP buffers is allocated, and if
they are full new SDEP buffers are allocated.

2. Fast Path builds log records for all DEDB and MSDB updates and for the input
and output messages; the records are passed to the logical logger. No forced
logging is requested, except in the case of sequential processing BMPs
(PROCOPT=P/H). MDSBs are updated and the locks are released. Virtual
Storage Option, VSO, DEDBs are updated (in the data space, not on DASD) and the locks are released.

If Phase 1 is unsuccessful, then the updated buffers are simply thrown away and the transaction is rescheduled (for IFP transactions), or a status code is returned to the program (for BMPs). The Fast Path Log Analysis utility reports on sync point failures. There is an entry for each transaction, whether successful or unsuccessful. The SYNC FAILURE column indicates the cause of the failure, for example:

- SDEP area is full
- FLD call verify fails
- Deadlock on a FLD call

At the end of sync point, nothing is physically logged, and updated DEDB CIs (non-VSO) are still in the buffers waiting on the OTHREADs and are locked.

**Post-Sync-Point Processing:** Eventually, the Fast Path log records are physically written to the log. This happens in either of two ways:

- The log buffer is filled by other IMS activity and written out to the online log datasets (OLDS).
- A timer pops, one-fourth of a second after the last log write. If new Fast Path transaction log records are in the log buffer unwritten, they are written out to the write-ahead datasets (WADS).

The key log record is the “sync point complete” type X ‘5937’ log record. The writing of this record to either the WADS or the OLDS triggers message sending and DEDB updating.

DEDBs are updated by system tasks called output threads. One OTHREAD is used for each DASD volume to be updated. If several control intervals in the same area data set are being updated, the buffer writes are chained and written with a single I/O. When MADS are involved, the chained writes are done to one ADS at a time by one OTHREAD. If the control intervals are in different areas on different DASD volumes, they are updated in parallel by multiple OTHREADs.

The locks acquired by the program against the updated CIs from non-VSO areas are held until the CIs have been written to the database. Accordingly, any program that tries to access these CIs is forced to wait until the OTHREAD has completed.
1.8 Performance Monitoring Information Sources

IMS is just part of the MVS complex. To fully understand its performance requires information about the total environment, as well as about IMS.

1.8.1 System Definition Data

The following list represents the data that is required to fully analyze a complete IMS MVS system; it is only a partial list of the different data that is ideally available.

1. A detailed CPU, DASD, and peripherals hardware diagram
2. A detailed network hardware diagram, including terminal line speeds

**Note**

Often, line speed is the major portion of online response time

3. A detailed system software list of all components used
4. DASD volume VTOC listings in formatted form for all active volumes
5. VSAM LISTCAT listings for all VSAM database data sets
6. The IMS procedures shown in Table 1

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Member Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control region JCL procedure</td>
<td>IMS</td>
</tr>
<tr>
<td>Control region parameter list</td>
<td>DFSPBxxx</td>
</tr>
<tr>
<td>Control region fixed pool parameter overrides</td>
<td>DFSSPMxx PROCLIB member</td>
</tr>
<tr>
<td>Control region fix list</td>
<td>DFSFIXxx</td>
</tr>
<tr>
<td>Control region database buffer pool definitions</td>
<td>DFSVSMxx</td>
</tr>
<tr>
<td>Control region preload list</td>
<td>DFSMPLxx</td>
</tr>
<tr>
<td>Dependent region JCL procedures</td>
<td>DFSMPR, IMSBATCH, IMSFP</td>
</tr>
<tr>
<td>Dependent region preload lists</td>
<td>DFSMPLxx</td>
</tr>
<tr>
<td>Dependent region program management</td>
<td>• SYS1.PARMLIB (CSVLLAxx)</td>
</tr>
<tr>
<td></td>
<td>• DFSVFLLxx</td>
</tr>
</tbody>
</table>

7. The MVS parameters shown in Table 2 on page 18
8. MVS pageable link pack area (PLPA) listing
9. IMS stage 1 system definition input listing
10. VTAM generation listing
11. NCP generation listing

### 1.8.2 IMS Full-Function Monitoring Data

Run the following for the IMS transaction peak load period:

- The IMS Monitor, tracing all options, for a 12 to 15 minute time period (see Figure 6). Use the same time period as the RMF Monitor I interval. RMF usually synchronizes the time to the hour, and it is more difficult to change this time period than Monitor’s.
  - Immediately before starting the IMS Monitor, take an IMS simple checkpoint (/CHE).
  - Immediately after stopping the IMS Monitor, take another IMS simple checkpoint (/CHE).

```
/CHE
/TRA SET ON MON ALL
   (trace interval)
/TRA SET OFF MON
/CHE
```

*Figure 6. Starting and Stopping the IMS Monitor*

- When executing the post processor program (DFSUTR20), print the DL/I CALL SUMMARY REPORT by specifying the statements shown in Figure 7 on page 19.
Figure 7. Printing the IMS Monitor Call Summary Report Using DFSUTR0

The distribution report is not required at this time and therefore need not be printed.

- An IMS /DIS POOL ALL command, entered both before and after the IMS monitor run.
- The IMSASAP II reports (Table 3), using the same data set created for the IMS monitor reports.

<table>
<thead>
<tr>
<th>Reports</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region summary</td>
<td>Always generated</td>
</tr>
<tr>
<td>Program summary</td>
<td>Always generated</td>
</tr>
<tr>
<td>IWAIT summary</td>
<td>Always generated</td>
</tr>
<tr>
<td>Transaction by time period</td>
<td>Time report</td>
</tr>
<tr>
<td>Region summary by region</td>
<td>Regions</td>
</tr>
<tr>
<td>Program summary by region</td>
<td>Regions</td>
</tr>
<tr>
<td>Region histogram</td>
<td>Histogram</td>
</tr>
<tr>
<td>Total system IWAIT detail</td>
<td>IWAIT(detail)</td>
</tr>
<tr>
<td>Enqueue/dequeue trace</td>
<td>Enqueue trace and DDname</td>
</tr>
</tbody>
</table>

- The IMSPARS reports (Table 4) for the 15 minutes monitored.

**note**

Specify a time period to IMSPARS that is about 10 seconds before and after the time stamps from the IMS simple checkpoints.

<table>
<thead>
<tr>
<th>Reports</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMSPARS start, stop</td>
<td>(mm/dd/yy.hh.mm.ss)</td>
</tr>
<tr>
<td>CPU usage</td>
<td>CPU(TRAN)</td>
</tr>
<tr>
<td>Internal resource usage</td>
<td>IRUR</td>
</tr>
<tr>
<td>Data set information</td>
<td>DBTR(TO(mm/dd/yy.hh.mm.ss))</td>
</tr>
<tr>
<td>Message queue utilization</td>
<td>MSGQ</td>
</tr>
<tr>
<td>Management exception report</td>
<td>MGRX</td>
</tr>
<tr>
<td>Transaction log report</td>
<td>LOG</td>
</tr>
<tr>
<td>User program supplied</td>
<td>USERPGM(USERPGM1)</td>
</tr>
</tbody>
</table>
1.8.3 IMS Fast Path Monitoring Data

Run the Fast Path Log Analysis utility (DBFULTA0) against the IMS log, specifying the same start and end time as for IMSPARS (spanning the two checkpoints). The reports clearly show the impact of the system checkpoints. Other recommended options (as SYSIN data) are:

- **NOT-MESSAGE** — to monitor all region types, not just IFPs
- **MAXDETAIL=1000** — to limit the size of the exception report
- **CALLS** — to get detailed information on DL/I calls
- **BUFFER** — to get detailed information on Fast Path buffer usage
- **VSO** — if using VSO DEDBs
- **TT(*)=ss.t** — where ss.t is the expected 90 percentile transit time

This last option causes detailed data to be produced for transactions (subject to MAXDETAIL value) that exceed the specified transit time (another name for time-in-system or internal response time). In Fast Path terminology, it is in-queue time + process time + out-queue time). Average values are reported for all MPP and IFP transactions.

**TT(*)** specifies that all transactions use the same criterion. If appropriate, you can also code specific time values for individual transactions.

1.8.4 MVS System Monitoring Data

Run the following RMF Monitor I reports for the 12 to 15 minute period:

- PROCESSOR ACTIVITY
- CHANNEL ACTIVITY
- DEVICE ACTIVITY - DASD
- PAGE/SWAP DATA SET ACTIVITY
- ENQUEUE/DEQUEUE REPORTS - DETAIL
- WORKLOAD ACTIVITY

Run the RMF Monitor II reports (Table 5) for the same period. Run them for a 15-minute period during the monitored interval with the options of DELTA,CYCLE(10S),INTERVAL(15M).

<table>
<thead>
<tr>
<th>Report Types</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address space state data</td>
<td>ASD</td>
</tr>
<tr>
<td>Address space resource data</td>
<td>ARD</td>
</tr>
<tr>
<td>Paging reports</td>
<td>SPAG</td>
</tr>
<tr>
<td>Real storage/processor/SRM</td>
<td>SRCS</td>
</tr>
</tbody>
</table>
1.8.5 Daily Summary

Run the IMSPARS reports (Table 6), using all (if practical) or a representative subset of the IMS logs created that day.

<table>
<thead>
<tr>
<th>Report Types</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU usage</td>
<td>CPU(TRANCODE)</td>
</tr>
<tr>
<td>Transaction transit time</td>
<td>ANALYSIS(TRAN)</td>
</tr>
<tr>
<td>Message queue utilization</td>
<td>MSGQ</td>
</tr>
<tr>
<td>Database information</td>
<td>DBTR(102/01/96,08.00.00)</td>
</tr>
</tbody>
</table>

1.8.6 Database Space Reports

The performance of transactions can be seriously impacted by disorganized or full databases. Therefore, it is important to have analysis reports of the databases available. Many customers produce these regularly as an extra step after each image copy, using the image copy data set as the source for the analysis.

The standard tools in this area are:

- **IMS System Utilities/Database Tools (DBT)**; product number 5685-093. For user who is on IMS V5 or later.

  or

- **IMS Database Analyzer V2**; product number 5665-349. For those still on IMS V4.
Chapter 2. IMS System Parameters and Implementation Options

This chapter discusses various system parameters and other aspects of the IMS implementation that relate to the performance of the system. Several major implementation choices, such as pool sizes, are omitted here and discussed at length in other chapters. The items covered in this chapter are:

- IMS simple checkpoints
- Dynamic allocation
- IMS/VTAM interface
- ESCON channels
- DASD fast write
- IMS logging

Warnings and recommendations are given as appropriate.

2.1 IMS Control Region Execution Parameters

This section describes some of the IMS execution parameters that have an effect on IMS performance.

2.1.1 EXVR Parameter

This parameter can be used to page fix the message queue buffer pool (EXVR=1). However, it is recommended that this parameter not be used. Instead, protect the message queue pool by means of the MVS storage isolation function (see 8.9, “Storage Isolation” on page 158), or page fix it using the DFSFIXxx member in PROCLIB, specifying POOLS=QBUF (see 8.7, “Page Fixing IMS Pools and Control Blocks” on page 153).

2.1.2 SRCH Parameter

This parameter is used if the IMS DL/I or interregion communication modules are moved to the MLPA of MVS and the control region has a STEPLIB or JOBLIB DD statement. If the SRCH parameter is set to 1, the IMS control region searches the MLPA before the JOBLIB or STEPLIB DD statement to find the IMS module being loaded. This can save virtual storage if multiple IMS TM systems or IMS batch (DL/I) jobs are executing concurrently in the same system. By referencing the modules in MLPA, they can be shared by all of the jobs rather than each job having its own copy. This is feasible only if all of the IMS systems are at the same release and service level.

The list of the IMS modules that can be loaded in MLPA is found in Table 12 on page 157.
2.1.3 PRLD Parameter

This parameter can be used to preload HDAM randomizing routines, segment edit or compression routines, and secondary index maintenance routines in the control region. The only reason to preload these modules is if the databases using them are frequently stopped during the online day by the /DBR command. This saves the physical load time when the databases are started again.

2.1.4 VSPEC Parameter

This parameter specifies which member (DFSVSMxx) in IMSESA.PROCLIB is to be used to define the VSAM buffer pools and various other IMS system options, many of which are related to performance. The DFSVSMxx parameters are these:

- **VSAM buffer pools, subpools, and hiperspace**

- **OSAM buffer subpools**

- **OSAM page-fixing options.**

- **VSAM insert option**
  Allow to default for an online system.

- **VSAMFIX**
  This parameter defines the VSAM pool page-fixing options. Refer to 8.7, “Page Fixing IMS Pools and Control Blocks” on page 153 for recommendations on the use of this option.

- **STRINGMX**
  This parameter specifies the maximum number of VSAM requests that can be processed concurrently. This value overrides the values obtained from the IMSCTRL MAXREGN parameter in IMS stage 1 and the PST= control region execution parameter (see 2.1.8, “PST Parameter” on page 26).

  Specify a value that is a little more than twice the maximum number of dependent regions that you expect to ever require.

- **BGWRT (VSAM Background Write Option)**
  This parameter specifies whether the background write (BGWRT) function for VSAM is to be activated and what percentage of the buffers are candidates for writing. The default is (YES,34); that is, BGWRT is on and with a percentage of 34.

  For further details see 11.3.3, “VSAM Background Write” on page 243.

- **ISSUE681 (BMP checkpoint message suppression option)**
  This is discussed in 12.4.1.3, “Checkpoint Message Suppression” on page 292.

- **VSAMPS Option**
  VSAMPS=LOCL is the only available choice, because IMS uses VSAM local shared resource pools.

- **OSAMOP IOSCB=NO**
  This is a batch option only and prevents OSAM facilities from being unnecessarily loaded into a batch address space (for programs that do not use OSAM databases).
• **OSAMOP OSAMGTF=YES**
  This again is a batch option to allow OSAM to write generalized trace facility (GTF) trace records.

• **SBONLINE**
  This option enables applications running in an online IMS system to use OSAM sequential buffering (SB). The actual use of OSAM SB is determined by the DFSCNTL DD statement in the dependent region JCL. By omitting the SBONLINE parameter, some virtual storage is saved, as the OSAM SB modules are not loaded.

• **IMS trace options**

• **DASD logging initialization parameters**

• **Coupling facility structure names** (for sysplex data sharing)

### 2.1.5 VAUT Parameter

This parameter specifies whether IMS is to use the VTAM authorized path facility. Using the VTAM authorized path requires less CPU for processing SENDs and RECEIVES, because VTAM performs less validity checking of the VTAM control blocks. Specify it as YES (VAUT=1); the default is NO.

### 2.1.6 LSO Parameter

This parameter specifies in which area of storage IMS is to allocate the database buffers, control blocks, and DL/I code. The options are these:

• **LSO=Y**
  This option allocates the storage associated with the database buffers, control blocks, and code in the control region private area. The log buffers, resident intent list, and enqueue/dequeue (ENQ/DEQ) storage are allocated in the control-region private area unless Fast Path is specified in the IMS system. In this case, the log buffers and ENQ/DEQ storage are allocated in CSA storage.
  
  This option results in a single PSB pool in extended CSA.

• **LSO=S**
  This option allocates DL/I storage in the DL/I separate address space. Database buffers and the ENQ/DEQ pool are in the DLS extended private area. Control blocks and DL/I code use private storage both above and below the 16 MB line.
  
  The PSB pool is in two parts, one in extended CSA (called PSBP) and the other in the extended private area of the DL/I separate address space (called DPSB).
  
  The log buffers and resident intent list are allocated from ECSA storage.
  
  The ENQ/DEQ pool is allocated in the DLS extended private area unless Fast Path is included in the system, in which case the ENQ/DEQ storage is allocated from ECSA.
  
  LSO=S is the recommended option in most cases, and certainly for production systems. IMS systems with a DL/I separate address space (LSO=S) generally use significantly less CPU than those using LSO=Y. When LSO=S, DL/I calls execute in cross-memory mode. When LSO=Y, DL/I calls must switch MVS
TCBs (from the dependent region to the control region), so this is a more expensive option.

An LSO=S system also generally experiences less virtual storage constraint than one using LSO=Y.

However, LSO=Y may be the best option for an IMS test system if the PSB and DMB pools and associated work pools are small and the system definition has a large number of transactions and databases defined. In this case, the amount of CSA storage required for LSO=S may exceed the amount of CSA storage required for LSO=Y. A virtual storage assessment must be done to determine which option to choose.

2.1.7 RES Parameter

To read an intent list from the ACBLIB requires an I/O. RES=Y causes all intent lists to be loaded at IMS startup, and so avoids ACBLIB I/O at program scheduling time. Additionally, any PSBs and DMBs defined as resident in IMS stage 1 are loaded and made resident in storage at IMS startup when RES=Y is specified. More detailed information can be found in 9.12, “Scheduling Pools” on page 188.

It is recommended that you use the default, RES=Y, in all production systems.

2.1.8 PST Parameter

The control block that represents a dependent region is called a partition specification table. (The IMS system address space also contain PSTs called System PSTs). The PST parameter is used to specify the minimum number of dependent-region PSTs to be created at IMS initialization time. If the number of dependent regions needs to be increased beyond this value, by either a /START REGION command or a BMP starting, IMS dynamically allocates a PST for each request up to a maximum of 999 (255 prior to IMS 5.1). When the task that requested the additional PST terminates, its PST is released. The number of PSTs never drops below the value in the PST parameter.

The PST parameter is not related to the number of dependent regions actually running, but it is usually approximately equal to the number of regions running in normal production mode.

If the STRINGMX parameter is not defined in the DFSVSMxx member (see 2.1.4, “VSPEC Parameter” on page 24), then the PST value is used to calculate the number of VSAM strings.

\[
\text{VSAM Strings} = (2 \times \text{PST}) + 1
\]

In this case, it is rather important to have the PST value at least as great as the maximum number of dependent regions. However, we recommend that the number of VSAM strings be defined explicitly in the DFSVSMxx member of PROCLIB.

2.1.9 ARC Parameter

This parameter specifies the number of OLDS that must be used and closed before automatic archiving can begin. The recommended specification for this parameter is ARC=1.
2.1.10 SAV Parameter

This parameter specifies the number of dynamic save area sets and prefixes (SAPs) for use by communications ITASKs. Communication delays result if this parameter is too small. As an order of magnitude, take the maximum number of active terminals and divide it by 15. The maximum value that can be specified is 999.

SAP shortages are reported by IMS monitoring (see 8.5.3, “Save Area Prefixes” on page 149).

2.1.11 Hash Tables (CHTS, LHTS, NHTS, UHTS)

Control block pools exist for CCBs, CNTs (LTERMs), VTCBs (VTAM nodes), and users (or SPQBss). The control blocks in these pools are not in any order. When IMS needs to locate one of these control blocks, it uses a hashing technique rather than performing a sequential search of the pool. Each pool has a hash table, and each hash table contains a user-specified number of slots or anchor points. The parameters of these hash tabales are known as CHTS, for conversation (CCB) hash table slots, LHTS, for LTERM hash table slots, NHTS, for terminal (VTCB) hash table slots, and UHTS, for user (SPQB) hash table slots.

When a control block is added to the pool, its name is hashed so that a number representing one of the hash table slots is generated (a concept familiar to anyone who uses HDAM). If that slot is not in use, the address of the new control block is put in it. If the anchor point is already in use, a synonym is created. Synonyms are chained together off the anchor point in ascending alphabetic order (just as HDAM roots form a synonym chain off a root anchor point).

For performance reasons, we recommend that control block synonym chains, on average, contain no more than seven blocks. Consequently, to calculate the number of required slots in each hash table, take the maximum number of resources that can ever exist (conversations, LTERMs, VTAM terminals, and users, respectively), divide by seven, and then round up to a power of 2.

For example, if you have a network with up to 18,000 ETO terminals expected to be concurrently logged on to IMS, then you need 18,000 VTCBs.

\[
18000 \div 7 = 2571
\]

next higher power of 2 = 4096

In this case, specify NHTS as 4096. LHTS would probably be the same (LTERMs), as would UHTS (users).

The default value (256 for all four parameters) is appropriate for up to 1792 resources (terminals). If you have significantly more than this number of terminals, set these parameters accordingly.

2.2 IMS Simple Checkpoint

The frequency with which IMS takes a simple checkpoint is defined on the IMS system definition IMSCTF macro statement as CPLOG=nnnn. This number can range from 500 to 16,777,215 and specifies the number of records to be written to the IMS log between simple checkpoints. The default value is 1,000, which is too low. The value to be specified is difficult to arrive at before the IMS system is in operation, but a starting value should be approximately 120,000.
The purpose of a simple checkpoint is to create a system restart point, so that control blocks and status information are written out to the log. Clearly, large systems generally take longer to checkpoint than smaller systems (all other things being equal), and there can sometimes be a short pause in transaction processing while a checkpoint completes. This is generally the result of all the log buffers filling up faster than they can be written to the OLDS. For possible solutions to this problem, see 9.20.3, “Log Buffer Tuning” on page 202.

If system checkpoints take a long time to complete (more than 5 seconds, for instance) and OLDS performance is known not to be the cause, then the problem is probably the result of excessive paging. Such problems are addressed in 8.4, “IMS Paging” on page 142.

If checkpoints are taken too frequently, there may be a noticeable impact on transaction processing. If checkpoints are not taken frequently enough, then the time required to perform an emergency restart is extended. An emergency restart normally reads the log records for the last checkpoint but one and then processes all subsequent log records.

A good value for the checkpoint frequency is one that causes a checkpoint every 10 to 15 minutes during the peak hour, although some customers do have significantly higher checkpointing frequencies. This is usually to ensure that one (or in some customers, two) checkpoints are taken on every OLDS. Some customers use automated operations to take a SNAPQ checkpoint at the start of each OLDS, and then set CPLOG to cause one more checkpoint on each OLDS. Taking at least one checkpoint (preferably two) on every OLDS is recommended for the following reasons:

- It optimizes system restart time.
- It minimizes the logs needed for database recovery with concurrent image copy.
- It minimizes the need for a tape restart (using SLDS).
- It provides running performance reports from the SLDSs.

Checkpoint performance monitoring is covered in 8.6, “IMS System Checkpoints” on page 151.

### 2.3 IMS Dynamic Allocation Considerations

Spikes in response time can occur whenever databases are dynamically closed by means of a /DBR IMS command or when a database is started (allocated) by means of a /START IMS command. This is the result of the dynamic allocation SVC being issued by the IMS control region or the DL/I separate address space.\(^5\)

Control is not returned to the IMS allocation task until the allocation is completed. A VSAM database opening could be delayed because the OPEN routine must also go to the VSAM catalog to acquire information about the database. This could take many seconds if there is contention for the MVS catalog.

---

\(^5\) DEDB area data sets are always allocated by the control region. Full function databases are allocated by the control region when using LSO=Y and by the DL/I separate address space when using LSO=S.
In a JES3 environment, a delay in response time could also result from a dynamic allocation being sent to the JES3 global processor and then returned to the local processor. Contention in the JES3 global processor could produce the delay.

Dynamic allocation is a major concern, because in most cases a block-mover latch is held. This can delay the scheduling of transactions and can show up as high block-mover busy time in the SCHEDULING and TERMINATION column on the IMS Monitor report.

**Data Set Recall:** One aspect of dynamic allocation that used to cause severe performance problems was when the requested database had to be recalled by HSM. As a result of the holding of a latch, transaction processing would stop until the recall was completed, and this would sometimes require an operator to mount a tape.

This problem is addressed in IMS/ESA 5.1. IMS issues a LOCATE call before requesting the allocation, and so discovers if the data set needs to be recalled. If a recall is necessary, IMS treats the database as unavailable and responds to the application that issued the DL/I call in the standard way:

- The DEEB call receives an FH status code.
- The DL/I DB call receives a BA status code if /INIT STATUS GROUPA is issued.
- Otherwise, the application is abended U3303 and the transaction is put on the suspend queue.

IMS issues the recall request under a separate subtask and periodically checks to see if it has completed. When IMS detects that recall has completed, dynamic allocation is attempted the next time a PSB that is sensitive to the database is scheduled, and the database is automatically restarted.

**VSAM Catalog Bottleneck:** A potential bottleneck still exists in waiting for the VSAM catalog. If any TSO user or batch job is enqueued for the catalog, IMS must wait until that task releases the catalog before it can complete the allocation of the database. This time could be milliseconds or seconds, and the transactions waiting for the databases must wait, along with any additional transactions of the same type on the IMS input queue.

The solution to this problem is to catalog the IMS databases in a user catalog and specify a STEPCAT DD statement in the IMS control region or the DL/I separate address space. This STEPCAT allows IMS to bypass searching the MVS master catalog when looking for the IMS database.

### 2.4 VTAM Tuning for IMS

The output of the /DIS A command provides some important information about the performance of the IMS VTAM interface. The lines labeled NODE IN and NODE OUT show the number of messages that are currently in process from VTAM to IMS (IN), or IMS to VTAM (OUT). These numbers are usually close to the transaction rate of the system. If the numbers are higher than the transaction rate, there could be an inefficiency in this process.

An example might be the specification of the OUTBUF parameter on the TERMINAL macro. If this parameter is specified as 256, and IMS is sending a
1000-byte message to VTAM, it takes four iterations of the IMS VTAM process to get the complete output message to VTAM. If many terminals are defined in this way, the number of messages in process tends to be high. The solution to this problem is to specify a larger value for the OUTBUF parameter.

An example of a problem with input processing could be with the storage allocation of the HIOP pool. If the pool does not have enough space for IMS to format the input message, IMS cannot release the RECANY buffer for a subsequent input message. VTAM stores the message in a data space reserved for this purpose. If this is the problem, the solution is to allow the HIOP to grow to a larger maximum size. This is done with the control region execution parameter HIOP=. However, we recommend that this parameter be left to default (to 2 GB - 1), so such problems are avoided.

Another possibility is that too few RECANY buffers are defined, although this is rarely a problem unless there is a dominant number of non-response-mode transactions that wait on input for DC log write-ahead. Again, the solution is simple; allocate more RECANY buffers (use the control region execution parameter RECA=).

Monitoring of HIOP (see 9.1, “IMS Buffer Pool Storage” on page 161) and RECANY usage (see 9.3, “RECANY Pool” on page 173) shows up such problems.

2.5 IMS and ESCON Channels

The ESCON channel subsystem offers high availability over long distances with flexible path switching. When used instead of a parallel channel, little difference in performance is experienced. Although the potential transmission rate of an ESCON channel (10 or 17 MB/second) is much faster than that of parallel channels (3 or 4.5 MB/second), total I/O response time and utilization of devices is very similar, channel transmission rate being a relatively small contributor. When used over long distances, ESCON channels do suffer a slight performance degradation (1 to 2 ms over 9 km) compared with standard channel-attached response.

The method of tuning channel, control unit, and string path is the same as for parallel channels, with performance data obtained similarly from the RMF Monitor I reports. The arrangement of channel paths and DASD strings is usually the same for ESCON as for parallel channels.

2.6 IMS Exploitation of DASD Fast Write

DASD Fast Write (DFW) performs write operations at cache speeds and has the potential for improving the performance of critical IMS data sets. Writing to DASD is not required to complete the DFW operation for write hits (including the so-called “predictable” format write operations). A copy of the data to be written is placed in cache and in nonvolatile storage (NVS), and the storage director returns channel-end and device-end. For write misses (when existing data is not being updated, or the write is not to the first new record on a track), physical access to the device is required.

The basic mode of caching is at the track level. In other words, the DASD control unit works with tracks of data in the cache rather than individual physical records.
Where databases are frequently updated, standard caching and DASD Fast Write can be very beneficial. Every block or CI that IMS has to update (including ISRT for an online system) must first be read, and so will be in the cache. Also, because IMS does its writes at sync point (or soon after for DEDB), there is very little chance of the data being removed from the cache before IMS does the write. So DL/I database updates always exploit DASD Fast Write.

If the ratio of writes to reads is low, however, basic cache management can be inefficient, especially where DL/I references only one or two records on a track.

IMS 5.1 supports dynamic cache management enhanced (DCME) for both VSAM and OSAM databases. This enables record-level caching to be used, which avoids the potential inefficiencies of track-level caching. The DASD subsystem monitors the use of the data and automatically switches between no caching, record-level, or track-level caching, according to which provides the best performance advantage.

One other benefit of using DCME is that caching is performed at the data set level, rather than at the DASD volume level.

To gain these full benefits from DCME, the database data sets must be managed by system-managed storage, and defined with the MAY CACHE attribute.

Certain IMS system data sets are ideal candidates for DFW:

- Log data sets
  The online data sets benefit from DASD Fast Write. Although not updating blocks, IMS uses format writes, which do exploit DFW.

  The write-ahead data sets also gain significant performance benefit from DFW. In a typically busy IMS system, IMS quickly cycles through the WADS tracks, so those tracks remain resident in the cache and always experience DFW.

  RAID5 DASD (for example, the IBM 9394) is optimized for read access. Consequently, it is essential to use DFW for write-intensive data sets such as WADS and OLDS.

- RECON data sets
  Although not used often (in comparison with other system data sets), when RECON data sets are used, a period of very intensive I/O is often experienced. At certain times, for example when many small batch jobs or image copies are running, the RECONs can be a potential bottleneck. Caching and DFW can provide significant performance benefit for these data sets.

2.7 IMS Logging

IMS has a sophisticated logging subsystem to which it logs all aspects of the system that require restart or recovery.

2.7.1 Online Log Data Set

IMS writes its log data to the OLDS. The user defines between 3 and 100 OLDS and, under the control of DBRC, the system cycles around the available copies. As each copy fills, it is scheduled to be archived by the Archive utility. The OLDS must reside on DASD, but are typically archived to tape.
2.7.1.1 Single or Dual OLDS Logging
OLDS logging can be either single or dual. If dual logging is used, true duality is supported; that is, IMS makes sure both copies are identical in content. For availability reasons, dual OLDS is recommended for production systems. For systems with an adequate number of log buffers, no performance penalty is incurred by having dual OLDS, because OLDS writes are asynchronous to transaction processing.

2.7.1.2 OLDS Buffers
We recommend that OLDS use half-track blocking. IMS uses the BSAM access method to write OLDS blocks. BSAM attempts to hold a single-buffer write request until the next one is issued, and then does a full track write. Under stress, IMS uses chained I/Os to write out multiple buffers at a time.

With LSO=S, the log buffers are in ECSA, and for the best performance we recommend that the number of OLDS buffers be at least 20 to 30. This helps to maintain performance during temporary spikes in the rate at which log records are created. These peaks can result from application abends (dumps), or more commonly, from simple system checkpoints.

When all of the log buffers are filled and waiting to be written out, no further building of log records is possible, and the system stalls. An increase in the number of log buffers reduces the likelihood of this happening.

The IMS log records are variable-length logical records, but they are written to the OLDS in fixed-length blocks. In other words, the last record in the block extends to the end of the buffer. Where possible, IMS accomplishes this by spanning the data to be logged across two log records, the first going at the end of one buffer, and the other starting the next. Otherwise, IMS pads out the log buffer with a type X'48' log record.


2.7.1.3 OLDS Allocation
The OLDS can be allocated on different DASD device types as long as the block size (BLKSIZE) is the same on all of the devices. The BLKSIZE must be a multiple of 2 KB with a minimum of 6 KB and a maximum of 30 KB. The recommended log buffer size is one-half track, or 26 KB for a 3390.

Put the primary and secondary OLDS (assuming dual logging) on different internal paths for best performance. In the case of a media failure, the remaining OLDS would still be available.

Do not allocate the OLDS on volumes or strings that are accessed by other systems, as the shared DASD contention could affect the OLDS I/O service time and, if all the log buffers fill up, lengthen the IMS transaction response time.
2.7.2 Archiving the OLDS

The Archive utility (DFSUARC0) executes as an MVS batch job and creates a system log data set (SLDS) and optionally, a recovery log data set (RLDS). The SLDS contains all log records, whereas the RLDS contains only those log records necessary for database recovery. The archive utility can also archive a batch DASD log to a SLDS or a RLDS on tape, for example.

An OLDS cannot be reused until the archive job has successfully completed. Flags are set in the DBRC RECON data set to signal when the OLDS needs archiving, when the archive job starts, and when the archive job completes successfully.

If the completion of the archive job is critical, specify a high dispatching priority. Assign a priority lower than the control region and the DL/I address space but higher than the dependent regions.

2.7.3 Write-Ahead Data Set

The WADS is the mechanism that IMS uses to provide high-performance log write-ahead. The WADS consists of preformatted 2080-byte blocks on DASD, with a key of 0.

Certain events, such as the full-function transaction sync point and MSC message logging, demand that log records be written out immediately. These forced log writes are done to the WADS, pending their being written to the OLDS when the log buffer fills up. Thus, the WADS contains the committed log records not yet written to the OLDS. The log buffer is considered to be a set of 2 KB pieces. At each forced write, IMS examines the current log buffer, and any 2 KB piece that contains new log data is written to the WADS. IMS selects a WADS track that does not contain previously committed pieces of this buffer, and writes the data (key 0) to the first 2080-byte record on the WADS that comes under the write head (or into the cache, when appropriate).

When the current OLDS buffer eventually fills up, it is scheduled to be written. Once the write completes, the WADS tracks containing that block can be reused.

With this technique, IMS actually needs a track for each 2 KB piece of a buffer, plus one. (For example, a 26 KB buffer needs 14 WADS tracks). This set of tracks is referred to as a track group. Because BSAM sometimes chains two buffer writes together, the WADS must have at least two track groups. This is the minimum requirement, but we recommend that the size of the WADS data set be defined large enough to have a track group for every log buffer. This way, if a number of OLDS are temporarily delayed, it is not possible to run out of WADS tracks when there are still log buffers available for transactions to use.

The use of dual WADS is recommended for some production systems only. With dual WADS, writes are done to both copies in parallel, so no performance penalty is incurred. However, dual writes use extra CPU cycles so it is not free. With today's hardware and operating system, it is highly unlikely that you would have both IMS and WADS DASDs failing at the same time. Therefore, dual WADS is a fairly expensive insurance policy. It is a user decision. The WADS is read by the online system only in the event of an emergency restart (for closing the OLDS).

---

6 WADS is also read by the DFSULTR0 utility when used to close the OLDS.
Allocate spare WADS data sets. If a write error occurs on a WADS, a new one can be switched to, and WADS processing continues. All the WADS must be allocated on the same device type with the same allocation.

If caching is not available, allocate the WADS on a lightly used device and string, as good WADS performance is necessary for a high performance transaction processing system.

If using RAID5 DASD (for example, IBM 9394), it is essential to use DASD Fast Write for the WADS data sets (see 2.6, “IMS Exploitation of DASD Fast Write” on page 30).

### 2.7.4 OLDS with XRF

An IMS extended recovery facility (XRF) places heavy demands on the logging process. The alternative IMS system reads all of the log records written by the active system; therefore, the I/O rate is effectively doubled. If the OLDS DASD response is continually delayed, the active system transaction response is delayed and the alternative system falls behind in its surveillance. This in turn causes an extended takeover time. Generally, the maximum logging rate of the XRF environment is one-half that of a non-XRF environment. The maximum logging rate for a single system is determined by the data rate available to the DASD device.
Chapter 3. Monitoring the IMS Environment

The effective use of system resources, CPU cycles, real storage, and I/O devices is the goal of any IMS installation. Clearly, it is necessary to be able to establish whether this goal is being met. If tuning is required, it is necessary to determine what specific performance problems exist in the system by examining the current use of the resources. When a potential solution is identified and implemented, it is necessary to reevaluate the use of the resources. Measurement tools that determine how the resources are used are necessary for any performance evaluation.

This chapter suggests a methodology for monitoring the IMS system. It looks at the tools that are recommended for monitoring the MVS environment in which IMS is running. The second part of this chapter explains how to use the associated monitoring reports, by extracting data and completing a set of worksheets that can be found in Appendix C, “Worksheets” on page 377.

Chapter 4, “Monitoring the IMS Subsystem” on page 61 discusses the tools for monitoring the IMS system itself and suggests how to use them to complete the remaining worksheets.

Chapter 5, “Identifying IMS Performance Problems” on page 109, then explains how to determine if and where any performance problems exist.

Chapters 6, 7, 8 and 9 examine in detail how to identify and understand these problems, broken down into the areas of CPU resource problems, I/O subsystem problems, storage and pool related problems, and problems related to the use of IMS-dependent regions. These chapters can also be used as general reference in these areas by anyone involved in IMS monitoring, performance evaluation, and tuning.

3.1 MVS Monitoring Tools

A considerable amount of data must be gathered in order to fully understand the behavior of the system and where a tuning effort can provide the best overall performance improvement. This section covers the tools most frequently used to gather data and describes how to use them to measure an IMS system. Familiarity with these tools and the data they provide is basic to any successful tuning effort.

3.1.1 MVS Data Collection Routines

Table 7 on page 36 lists the MVS tools that are required to perform an IMS MVS tuning exercise.

Two data collection routines are integrated into the MVS system: system management facility (SMF) and generalized trace facility. GTF collects huge amounts of detailed trace data, but no easy-to-use tool is available for analyzing the data and producing reports. GTF records can be individually listed with IPCS.

Several program products used in the IMS/ESA MVS environment can provide additional data. Resource management facility (RMF) measures system activity and collects this data in the form of SMF records and SYSOUT listings. Advanced communication function/Virtual Telecommunications Access Method (ACF/VTAM)
Table 7. MVS Data Collection Routines Summary

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Description</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Measurement Facility (RMF)</td>
<td>Program Product</td>
<td>Collects systemwide data about CPU, channel, I/O device, paging activity,</td>
<td>RMF User's Guide; GC33-1038 for SP 4</td>
</tr>
<tr>
<td></td>
<td>5685-029 for MVS SP 4</td>
<td>SRM/WLM workload activity. It also provides data associated with each active address space to give a full picture of all activity in the system.</td>
<td>GC33-6483 for SP 5</td>
</tr>
<tr>
<td></td>
<td>5655-084 for MVS SP 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache RMF Reporter</td>
<td>Program Product</td>
<td>Automatic data collection from RMF and a postprocessor batch report program.</td>
<td>Cache RMF Reporter (CRR) PDO (SH20-6295)</td>
</tr>
<tr>
<td></td>
<td>5798-DQD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized Trace Facility (GTF)</td>
<td>MVS component</td>
<td>Collects system-related events by tracing system events. No report program is supplied with the product.</td>
<td>MVS/ESA SP V5 Initialization and Tuning Reference (SC28-1452)</td>
</tr>
<tr>
<td>Service Level Reporter (SLR)</td>
<td>Program Product</td>
<td>Collects systemwide data from SMF and subsystems log, summarizes for operation management.</td>
<td>SLR User's Guide (SH19-6530)</td>
</tr>
<tr>
<td></td>
<td>5665-397</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SystemView Enterprise Performance Data Manager (EPDM): IMS Feature</td>
<td>Program Product</td>
<td>Collects systemwide data similar to RMF and IMS log data, summarizes the data, and presents it in a variety of forms for use in systems management.</td>
<td>EPDM IMS PF; Guide &amp; Reference (SH19-6825)</td>
</tr>
<tr>
<td></td>
<td>5695-101</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.1.1 Resource Management Facility

RMF is a separately orderable product in the MVS/ESA environment. It usually coincides with MVS release. RMF V4R3, program product 5685-029, for example, is for the MVS SP 4 (MVS/ESA 4.2.2 and 4.3) environment, and RMF V5R2, 5655-084, is for MVS SP 5 (MVS/ESA 5.1.0, 5.2.0 and 5.2.2). RMF becomes an OS/390 base product and comes with OS/390 system. All major RMF report layouts look identical, regardless of version. Later version of reports may have more details on data collection. However, the parameters shown for various reports can be interpreted differently on different releases. Since IMS V5 can be run from MVS/ESA V4.2 up to the latest OS/390, the reader should examine parameter definition between releases carefully before drawing conclusion for the result. The sample output and parameter interpretations in this book are the output...
of RMF V5R2 under MVS/ESA SP5.2.0. Refer to RMF and product manuals for detailed descriptions of parameters if you are using a different release of RMF or operating system.

RMF collects systemwide data that describes the CPU activity, I/O activity (channel and device usage), real storage activity (demand and swap paging statistics), and system resource manager/workload manager (SRM/WLM) workload activity. Keep RMF active in the system 24 hours a day, and run it at a dispatching priority above that of other address spaces in the system. If you do, reports are written at the requested intervals and other work is not delayed because of locks held by RMF.

Reports are generated at the time intervals specified by your installation. The largest system overhead of RMF occurs during report generation; the shorter the interval between reports, the larger the burden on the system. An interval of 10 to 30 minutes is recommended for normal operation. To address a specific problem, reduce the time interval to 1 to 10 minutes. The RMF records can be directed to the SMF data sets, using the options NOREPORT and RECORD. The SYSOUT report overhead is not incurred, and the SMF records can be formatted at a later time.

Although three types of RMF sessions can be executed only two are discussed:

- RMF Monitor I session provides data on systemwide resources such as paging activity, channel activity, device activity, SRM or WLM workload activity, and enqueue contention. The data is used to determine how well the overall system is performing.
- RMF Monitor II session collects information about each address space in the system, as well as paging, real storage, processor, and SRM or WLM activity. Monitor II is basically a snapshot type of session because it generates reports from a single sample of system indicators.

In addition to the traditional batch reports, RMF information is typically displayed in a MVS TSO/ISPF session. In the OS/390 environment, however, a new RMF client/server enabling (RMFCS) function is now available. This new function allows RMF display on an OS/2 workstation without TSO session. Furthermore, it allows establishing as many sessions as needed with all those MVS systems in the network that have an APPC or TCP/IP connection configured to the workstation. It acts as a single point of control for all performance management. Refer to OS/390 RMF manuals for detail on this new function.

These types of information are displayed on the RMF Monitor II reports:

- MVS dispatching data
  - Dispatching priority
  - Dispatching position in the queue
- Working set size (real storage usage)
- Start I/O rate per second in each address space
• Paging rates per second in each address space:
  – SWAP INs + OUTs
  – LPA (PLPA INs)
  – CSA (COMMON INs + OUTs)
  – PRIVATE AREA INs + OUTs
  – VIO INs + OUTs
• CPU time
  – Total mode or delta mode
  – TCB CPU time only
  – TCB + SRB CPU time
• System storage utilization
  – SQA fixed frames
  – LPA pageable frames
  – LPA fixed frames
  – CSA pageable frames
  – LPA + CSA fixed frames
  – LSQA fixed frames
  – Private fixed frames

3.1.1.2 Virtual Telecommunication Access Method
ACF/VTAM provides information about buffer usage either to GTF, using SMS trace
data, or to the system console through a DISPLAY command. Other tuning
statistics may also be recorded on the system console through the
MODIFY proname, using the TNSTAT command. (This command is described
in the VTAM V4R3 Diagnosis LY43-0069)

3.1.1.3 Service-Level Reporter
SLR gathers transaction response time, database access time, and the like from
SMF as well as IMS logs, summarizes them and compares them to a policy
(service level) database to determine IMS availability. It is a high level reporting
and monitoring system for production management.

3.1.1.4 Enterprise Performance Data Manager
EPDM comes in several components. Its base system collects MVS system data
and stores it onto a set of DB2 tables. Its IMS feature component gathers
additional information from IMS logs. It is an IBM SystemView component and
eventually will replace part of RMF functions and SLR in a future OS/390 release.
3.2 The MVS Workload Manager

The MVS WLM was introduced in MVS Version 5. It provides the option of defining work management objectives in terms of performance goals, and it allows different types of work to be classified. Each service classification has a performance goal combined with a measure of its importance to the business.

When resource demands are less than the resources available, all work processes at its normal rate. However, when resources (CPU or main storage) are in contention, the WLM, based on defined goals and their relative importance, dynamically adjusts the work in an attempt to achieve all of the goals.

3.2.1 WLM and IMS

The WLM allows IMS transactions to be defined in different service classes. As a transaction is running, IMS supplies the WLM with performance measurements to help it provide the necessary system resources. To this end, the WLM is told about and understands the critical importance of the IMS service address spaces (control region, DLISAS, DBRC, and IRLM).

IMS supplies the WLM with performance data in two ways: it directly reports the internal response time of each transaction, and it also provides indirect information by updating a performance block throughout the life of each transaction. The performance block indicates the state of the transaction; for example, executing, waiting on I/O, or waiting on a lock. The WLM periodically (at best, a few times per second) looks at the performance block to determine what is delaying the transaction. It uses a sampling technique to minimize the overhead of getting the delay data.

MVS allocates CPU and storage resources to address spaces, not to individual transactions. So for the WLM's monitoring technique to be effective, all transactions going through a dependent region must be of roughly equal profile, be of equal value to the business, and have the same performance goals.

The way to achieve this is to define WLM service classes based on transaction scheduling classes and to avoid mixing transactions of different types in the same region.

These aspects of the WLM are covered in more detail in the ITSO redbook IMS/ESA Version 5.1 Guide (GG24-4302).

In large IMS production systems, where IMS is the main user of the MVS system, the purpose of the WLM is to allow IMS as much resource as it can use and prevent other lower priority work from impacting the IMS performance. In this case, be sure to define very aggressive performance goals for the IMS system. For example, if a certain class of transaction normally has an internal response time of 0.25 second, then define the corresponding goal as 0.1 second.

3.2.2 WLM and RMF Transaction Monitoring

RMF can be used to report on IMS performance relative to the goals. In the same way that you can have service classes, you can also have reporting classes. In general, define the RMF reporting classes the same as the service classes.
3.2.3 WLM and the Performance Analyst

RMF transaction monitoring in no way replaces or reduces the importance of the more traditional monitoring described in this book. Similarly, the WLM itself does not work magic: it does not remove the need to monitor and tune the MVS environment, the IMS subsystem, the I/O subsystem, or the databases.

What the WLM does, and does so very well, is make it easy to define clear and user-friendly performance objectives. It is also excellent at managing the system resources to optimal effect. However, it cannot prevent conflicts, contentions, and overcommitments; it can only attempt to manage them.

It is still necessary for you to identify the conflicts and contentions, the reasons why the WLM cannot meet its objectives. Look for adverse trends, investigate the causes, identify bottlenecks, and recommend remedial action.

3.3 Analysis of Environment Data

Once all of the measurements are complete and the reports are generated, the data must be analyzed to find the key indicators that identify the performance problems, if any, within the system. This section shows which data to consider and how to interpret that data in the performance evaluation.

3.3.1 CPU Resource Use

CPU resource use addresses the utilization of available CPU processing cycles, which jobs are processed, and in what priority they are processed. The reports used in analyzing the CPU workload are the RMF Monitor I CPU Activity report, the RMF Monitor II Address Space State Data (ASD) report, and the RMF Monitor II Real Storage/Processor or SRM Activity (SRCS) report.

Execute the RMF reports in delta mode, so that the delta times do not have to be calculated manually. In delta mode, RMF calculates the amount of CPU and task control block (TCB) time used in the last interval. If delta mode is not specified, the CPU and TCB times are the total time since the address space was started.

Appendix C, “Worksheets” on page 377 and Figure 128 on page 378 provides a facility for collecting RMF data for further study.

3.3.1.1 General Data

We recommend that you record the average CPU utilization from the RMF Monitor I CPU Activity report. For example, in Figure 8 on page 41, the average CPU utilization over the 15-minute period is 86.94%. This is the average; it does not tell you about any temporary peaks or valleys during the interval. The RMF Monitor II SRCS report allows further analysis of what occurs in the CPU during the RMF reporting interval. CPU overcommitment can be readily seen in the RMF Monitor II SRCS report (Figure 9 on page 42), under the heading CPU UTL. Scan the CPU column to see the different percentages of CPU utilization over the time interval. Values over 100% suggest that care be taken to ensure that IMS is getting the appropriate proportion of the CPU resource.
### 3.3.1.2 Dispatching Priority

Dispatching priority within an MVS system, when running in compatibility mode, is determined by the installation performance specification (IPS) member that is in effect. The workload manager, operating in goal mode, sets and dynamically adjusts priorities as required to meet the goals.

The MVS dispatcher queue sequence number indicates the order in which MVS services the address spaces. The address space with the lower sequence number receives CPU service first when it has work to do. The address space dispatcher queue sequence number is determined by its dispatching priority relative to other swapped-in address spaces: the higher the relative dispatching priority, the lower the sequence number.
Address spaces with the same dispatching priority, except for those in the mean-time-to-wait priority group, are assigned sequence numbers in the order in which they are swapped-in; the last one in has the lowest priority. The dispatcher queue sequence number changes when a task with a higher dispatching priority is swapped in or out. This could be the result of swapping an MVS Initiator or a TSO address space in or out.

Monitoring the environment requires that you identify which address spaces are getting the CPU resource. The RMF Monitor II ARD report (see Figure 20 on page 60) lists all the address spaces and how much CPU each uses (column headed CPU TIME). Use this report to identify the major users of CPU. Then use the RMF Monitor II ASD report (Figure 10 on page 43) to find the attributes of these address spaces and record them on the worksheet. The key columns are: dispatching priority (DP/PR) and current location (C/L). Current location indicates whether the address space is swappable (NS=nonswappable, IN=in storage, OT=swapped out and ready, >=being swapped out). Other columns indicate the number of central storage (CS) and expanded storage page frames (ESF) in use, whether the address space is predominantly cross-memory such as DLISAS (X/M), and paging activity (PIN RT and ES RT).

DB2CIRLM has a dispatching priority of 5B. This would result in DB2 issuing DSNE405E message.

### 3.3.2 I/O Resource

Problems reported as poor IMS response can often be traced to I/O subsystem contention for the channel, control unit, or DASD head of string. This problem usually resides in a shared DASD environment, but can reside in a nonshared DASD environment as well. As a general tuning rule, the I/O subsystem is one of the first areas to examine to determine whether it is the cause of the poor response.

The utilization of the system channels and DASD devices is the major concern in the area of I/O resources.
3.3.2.1 General Data
Knowledge of the hardware configuration, DASD volume placement, DASD mount, and shared or nonshared attributes is necessary. List the volume serial number and unit address for each DASD volume that contains IMS data sets. Determine from the unit address if the IMS volume is on a control unit that is shared with another system. If it is shared, also note which other volumes reside on the same control unit.

The output from an MVS console DISPLAY UNIT,DASD,ONLINE command indicates whether the IMS volumes are mounted as PRIVATE, as opposed to PUBLIC or STORAGE. The mount attribute of PRIVATE is important, as it prevents device contention and the DASD arm movement that can result if any work data sets are allocated on the IMS volumes.

Use Figure 129 on page 380, Appendix C, “Worksheets” on page 377 to record information about the IMS DASD volumes.

3.3.2.2 Channels
The RMF Monitor I Channel Path Activity report (Figure 11 on page 45) lists all channels and the percentage of time each channel path is busy. Any channel on the report that is more than 30% busy requires further study.

Using the hardware configuration, note which of the channels are DASD channels and which volumes are on each channel. Tape channels can be ignored, as they do not affect the performance of the online IMS system. However, tape channels can affect the log archive process, so monitor them if they affect the archiving of the DASD logs.

List any problem channels on the worksheet copied from Appendix C.
3.3.2.3 Direct Access Storage Devices

The key to finding and eliminating I/O-related performance problems is to analyze I/O response time. I/O response time is the time it takes to complete an I/O operation. This elapsed time has a dramatic effect on performance. The discussion that follows addresses the various service time elements and the factors that can affect them.
## Channel Path Activity

**MVS/ESA**
**SYSTEM ID** SYS1
**DATE** 03/14/96
**RPT VERSION** 5.2.0
**TIME** 16.30.00
**INTERVAL** 14.59.946
**CYCLE** 1.000 SECONDS

**IODF = A3**
**CR-DATE:** 03/02/96
**CR-TIME:** 07.42.20
**ACT:** POR
**MODE:** LPAR
**CPMF:** AVAILABLE

### Channel Path Activity Report

<table>
<thead>
<tr>
<th>Channel Path</th>
<th>Utilization (%)</th>
<th>Channel Path</th>
<th>Utilization (%)</th>
<th>Channel Path</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Type</td>
<td>SHR</td>
<td>Partition</td>
<td>Total</td>
<td>ID</td>
</tr>
<tr>
<td>00</td>
<td>BL</td>
<td>OFFLINE</td>
<td>08</td>
<td>BL</td>
<td>0.00</td>
</tr>
<tr>
<td>01</td>
<td>BL</td>
<td>OFFLINE</td>
<td>09</td>
<td>BL</td>
<td>1.03</td>
</tr>
<tr>
<td>02</td>
<td>BL</td>
<td>6.62</td>
<td>6.62</td>
<td>0A</td>
<td>BL</td>
</tr>
<tr>
<td>03</td>
<td>BL</td>
<td>18.95</td>
<td>18.95</td>
<td>0B</td>
<td>BL</td>
</tr>
<tr>
<td>04</td>
<td>BL</td>
<td>0.17</td>
<td>0.17</td>
<td>0C</td>
<td>BL</td>
</tr>
<tr>
<td>05</td>
<td>BL</td>
<td>3.08</td>
<td>3.08</td>
<td>0D</td>
<td>BL</td>
</tr>
<tr>
<td>06</td>
<td>BL</td>
<td>0.00</td>
<td>0.00</td>
<td>0E</td>
<td>BL</td>
</tr>
<tr>
<td>07</td>
<td>BL</td>
<td>0.06</td>
<td>0.06</td>
<td>0F</td>
<td>BL</td>
</tr>
<tr>
<td>18</td>
<td>BL</td>
<td>0.00</td>
<td>0.00</td>
<td>20</td>
<td>BL</td>
</tr>
<tr>
<td>19</td>
<td>BL</td>
<td>5.99</td>
<td>5.99</td>
<td>21</td>
<td>BL</td>
</tr>
<tr>
<td>1A</td>
<td>BL</td>
<td>0.54</td>
<td>0.54</td>
<td>22</td>
<td>BL</td>
</tr>
<tr>
<td>1B</td>
<td>BL</td>
<td>0.00</td>
<td>0.00</td>
<td>23</td>
<td>BL</td>
</tr>
<tr>
<td>1C</td>
<td>BL</td>
<td>0.03</td>
<td>0.03</td>
<td>24</td>
<td>BL</td>
</tr>
<tr>
<td>1D</td>
<td>BL</td>
<td>0.01</td>
<td>0.01</td>
<td>25</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>1E</td>
<td>BL</td>
<td>9.30</td>
<td>9.30</td>
<td>26</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>1F</td>
<td>BL</td>
<td>OFFLINE</td>
<td>27</td>
<td>BL</td>
<td>4.68</td>
</tr>
<tr>
<td>30</td>
<td>CN D</td>
<td>OFFLINE</td>
<td>38</td>
<td>CV</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>31</td>
<td>CN D</td>
<td>OFFLINE</td>
<td>39</td>
<td>CV</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>32</td>
<td>CN D</td>
<td>OFFLINE</td>
<td>3A</td>
<td>CN D</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>33</td>
<td>CN D</td>
<td>4.21</td>
<td>4.21</td>
<td>3B</td>
<td>CV</td>
</tr>
<tr>
<td>34</td>
<td>CV</td>
<td>OFFLINE</td>
<td>3C</td>
<td>CV</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>35</td>
<td>CV</td>
<td>0.94</td>
<td>0.94</td>
<td>3D</td>
<td>CV</td>
</tr>
<tr>
<td>36</td>
<td>CV</td>
<td>16.43</td>
<td>16.43</td>
<td>3E</td>
<td>CV</td>
</tr>
<tr>
<td>37</td>
<td>CV</td>
<td>4.33</td>
<td>4.33</td>
<td>3F</td>
<td>CN D</td>
</tr>
</tbody>
</table>

**Figure 11. RMF Monitor I Channel Path Activity Report**
Refer to Figure 12 on page 48, where all times are reported in milliseconds:

- **IOS Queue Time (AVG IOSQ TIME)**
  This is the time an I/O request must wait on an IOS queue before a start subsystem channel (SSCH) instruction can be issued. This delay occurs when a previous request to the same device or subchannel is in progress.
  This time also includes the time waiting because the DASD is reserved by another system. The first SSCH ends with a status code, causing MVS to queue the I/O request until notified to retry. This gives MVS the opportunity to apply I/O priorities if more than one I/O request is queued.

- **PENDING Time (AVG PEND TIME)**
  This is the time an I/O request must wait in the channel subsystem after the SSCH is accepted and before the first CCW is accepted by the DASD subsystem. This delay occurs when a previous request to the same device from another system is in progress or the channel, control unit, or head of string is busy.

- **DISCONNECT Time (AVG DISC TIME)**
  This is the time an I/O request is disconnected from the channel path during an I/O operation. This time represents the time the device is in use but not transferring data to this processor (for example, seek and set sector time).

- **CONNECT Time (AVG CONN TIME)**
  This is the time a device is connected to a channel path to search for or transfer data between the device and real storage.

- **I/O Response Time (AVG RESP TIME)**
  This is the elapsed time from the execution of the first SSCH issued for the device to the completion of the data transfer (channel end or device end interrupt). The time includes all hardware and the input/output supervisor software queue time, plus the actual I/O operation.

Path contention caused by channel busy, control unit busy, and reserves by other systems can be a major cause of I/O queuing. This shows as IOSQ time and PENDING time.

Queuing caused by shared DASD or string switching is reported by RMF as AVG DISC TIME and AVG IOSQ TIME. This time can be elongated if the head of string is busy transferring data. MVS does not distinguish the busy state due to shared DASD from the busy state that results when the head of string is connected to another channel on the same CPU.

I/O response time can be monitored in two other reports:

- IMS monitor REGION IWAIT report
- IMS monitor PROGRAM I/O report.

These IMS monitor reports indicate the I/O response time but also include any time spent waiting to be dispatched by MVS because higher priority tasks are executing.

The RMF direct access device activity report (Figure 12 on page 48) offers two ways to determine whether a DASD is causing poor response:
1. The Percent Device Utilized indicates the percentage of time during which an outstanding I/O request was issued to the device. Any number greater than 50% is considered a problem.

2. If a device has an IOSQ Time greater than 4 ms, an Average Pend Time greater than 4 ms, or an Average Disconnect Time greater than 16 ms, it should be examined to determine if it is too busy.

A high Device Activity Rate is a warning of possible future problems. An activity count is considered high only when viewed in relation to the activity counts of devices that have high IOSQ time or a high percent busy. Thus, a device can have a high activity count but may not be overused in terms of queue length and percent busy.

For example, consider Figure 12 on page 48. With typical average response times of 2 to 4 ms and disconnect times (which includes seek time) of about 0, this report is for cached DASD. Looking at volume SYSLBX (address 01AE), which is averaging over 125 I/Os per second with a response time of 2 ms, we see that connect time is 1.4 ms, and the device utilization is a comfortable 18%.

In contrast, look at volume PGTIB5 (address 01B5). This is performing only 2.166 I/Os per second, but the average response time is 23 ms, mainly consisting of 13.2 ms disconnect time and 9.6 ms connect time. This volume is probably uncached and performing half-track or chained I/Os.

Also look at volume SLR005 (address 01BC). This is performing only one I/O every 8 seconds (device activity rate of 0.124). The response time is 11 ms, of which 9 ms is IOSQ time. A clue to a possible cause is seen in the column % DEV RESV. This shows that the volume was reserved by this system for 0.1% of the time. A reserve on this system does not cause delays, but it does suggest the possibility that another system is also using RESERVE.

Using the Appendix C worksheet (Figure 129 on page 380), record the IMS volumes that exceed the criteria listed. The reason code is a number 1 through 5, depending on which criterion has been exceeded.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SGLARG90 019B 33902</td>
<td>SML022</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A0 33902</td>
<td>SML015</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A1 33902</td>
<td>SML025</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A2 33902</td>
<td>SML016</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A3 33902</td>
<td>SML028</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A4 33902</td>
<td>SML017</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A5 33902</td>
<td>SML029</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A6 33902</td>
<td>SML018</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLARG90 01A7 33902</td>
<td>SML030</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>01A8 33902</td>
<td>TSOL63</td>
<td>000F</td>
<td>0.003</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>8.7</td>
</tr>
<tr>
<td>01A9 33902</td>
<td>TSOL64</td>
<td>000F</td>
<td>0.153</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>11.4</td>
</tr>
<tr>
<td>01AA 33902</td>
<td>SCR001</td>
<td>000F</td>
<td>0.024</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.1</td>
</tr>
<tr>
<td>01AB 33902</td>
<td>SCR002</td>
<td>000F</td>
<td>0.037</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.1</td>
</tr>
<tr>
<td>01AE 33902</td>
<td>SYSL8X</td>
<td>000F</td>
<td>125.067</td>
<td>2.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>6751</td>
</tr>
<tr>
<td>01AF 33902</td>
<td>TSOL06</td>
<td>000F</td>
<td>0.232</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>5.2</td>
</tr>
<tr>
<td>SGLSR 01B4 33903</td>
<td>SRL001</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>100.0</td>
</tr>
<tr>
<td>01B5 33903</td>
<td>PGT1B5</td>
<td>000F</td>
<td>2.166</td>
<td>23.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>4.95</td>
</tr>
<tr>
<td>SGLSR 01B6 33903</td>
<td>SRL002</td>
<td>000F</td>
<td>0.008</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>2.4</td>
</tr>
<tr>
<td>01B7 33903</td>
<td>PGT1B7</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>2.4</td>
</tr>
<tr>
<td>SGLSR 01B8 33903</td>
<td>SRL003</td>
<td>000F</td>
<td>0.020</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>2.4</td>
</tr>
<tr>
<td>01B9 33903</td>
<td>HSM0CD</td>
<td>000F</td>
<td>0.206</td>
<td>3.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLSR 01BA 33903</td>
<td>SRL004</td>
<td>000F</td>
<td>0.081</td>
<td>4.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>01BB 33903</td>
<td>SPR1BB</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLSR 01BC 33903</td>
<td>SRL005</td>
<td>000F</td>
<td>0.124</td>
<td>11.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>01BD 33903</td>
<td>SPR1BD</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>SGLSR 01BE 33903</td>
<td>SRL006</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>01BF 33903</td>
<td>SPR1BF</td>
<td>000F</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.0</td>
</tr>
</tbody>
</table>

LCU: 000F 173.697 3 1 0.0 0.0 0.0 0.3 0.2 1.7 0.56 0.69 0.2 7233 100.0 0.0
3.3.2.4 Control Units and String Switches
Each DASD shares a control unit and a head of string or internal path with other DASDs. If the control unit, head of string, or internal path is busy, the I/O request to a nonbusy device on that path results in a busy condition. Control units or strings that are overly busy increase wait times for I/O.

The control unit is busy during data transfer and is connected to the channel during that time. If only one channel is connected to the control unit, no control-unit-busy is expected, as the channel would have been busy and the SSCH would have been queued. If an alternate channel is connected to that control unit, or another CPU has a channel connected to that control unit, there could be an excessive number of control unit busy occurrences caused by either too many devices on the control unit or one device that dominates the control unit and string. This dominant volume is easy to identify by the number of SSCHs per second issued to it. If the cause is not a dominant volume, then further investigation is needed.

The 3390 has up to four internal paths to each string from its control units. Arrange the volumes on each string to isolate the active volumes on a different internal path from the IMS volumes on that same string.

3.3.2.5 DASD Seek Patterns
Although the channel is disconnected during the seek or set sector operation of a DASD I/O, the device is still considered to be busy. Wide seek patterns generate longer I/O times and device contention. Therefore, it is important to alleviate wide seek patterns wherever possible. They show up as long disconnect times.

3.3.2.6 ESCON and DASD Fast Write
The ESCON channel subsystem provides high availability. Performance tuning with ESCON is the same as with parallel channels. RMF fully supports an ESCON channel subsystem. Channels, DASD, control units, and string switches are reported in the same way as for an environment of parallel channels.

DASD Fast Write (DFW) can improve IMS performance. DFW hit ratios can be analyzed using the Cache RMF Reporter. A sample report is shown in Figure 13 on page 50. This report shows the dominant volumes and clearly indicates the benefits provided by caching, including DASD Fast Write.

Various considerations for using DFW with IMS are described in 2.6, “IMS Exploitation of DASD Fast Write” on page 30.

3.3.2.7 I/O Subsystem Tuning
IBM publishes several documents on DASD tuning. Most of the recommendations in these documents pertain to IMS as well as batch or TSO. Therefore, this book does not describe tuning the DASD subsystem. Relevant publications in this area include:

- 3990/9390 Storage Control Introduction (GA32-0098)
- 3990/9390 Planning, Installation and Storage Administration; GA32-0100
- 3990 Model 3 ESCON Performance Measurements (GG66-3233)
- 3990 Model 3 DASD Fast Write Extended Function Performance Measurement; GG66-3160
3.3.3 Real Storage and Expanded Storage Resource

Real and expanded storage overcommitment on an IMS system causes an increase in IMS response time as the system paging rate and page movements increase. The RMF Monitor I Paging Activity report and the RMF Monitor II ARD report are used to determine the amount of paging activity in the system.

A sample worksheet is provided on page 382 for collecting RMF data for further study.

Figure 13. Caching Subsystem Summary Report
3.3.3.1 Storage Utilization

The MVS pageable pool is the total number of page frames of real storage and expanded storage in the system, minus the number of page frames that are long-term fixed in real storage. The reason for calculating the number of frames in the pageable pool is to show the actual amount of real and expanded storage available for use by the working sets of the address spaces. When the pageable pool is overcommitted, demand paging results.

The RMF Paging Activity report (see Figures 14 through 16 on pages 52 - 54) is used for the calculations of real and expanded storage that follow. The headings referred to in the following descriptions can be found under the heading FRAME AND SLOT COUNTS on page 53. The AVG column is used for these calculations.

**MVS Nucleus:** The heading Nucleus under Fixed Frames specifies the number of nucleus frames that are fixed in this system. The number shown on the report is 3004.

**Number of System Queue Area Fixed Frames:** The heading SQA under Fixed Frames specifies the number of SQA frames that are fixed in this system. The number shown on the report is 5598.

**Number of Link Pack Area Fixed Frames:** The heading LPA under Fixed Frames specifies the number of LPA frames that are fixed in this system. The number shown on the report is 168.

**Number of Common Fixed Frames:** The heading CSA under Fixed Frames specifies the number of CSA frames that are fixed in this system. The number shown on the report is 1732.

**Number of Local SQA Fixed Frames:** The heading LSQA under Fixed Frames specifies the number of LSQA frames that are fixed in this system. The number shown on the report is 3099.

**Private Fixed Frames:** The heading Regions+SWA under Fixed Frames specifies the total number of private area frames that are fixed in the system. The number shown on the report is 703.

**Pageable Pool Page Frames:** The headings Total Frames under Central Storage and Expanded Storage specify the number of pageable frames available to the system. The numbers shown on the report are 129,248 and 262,634, yielding a total of 391,882.

**Total Fixed Frames:** The heading Total Frames under Fixed Frames specifies the total number of fixed frames in the system. The number shown on the report is 13,632 (about 55 MB).

By itemizing these fixed frames, you can determine whether any reduction in fixed storage can be accomplished in any areas. It might also be a benefit at a later time if paging increases. A comparison could be made to determine whether additional pages were fixed and, if so, in what area.
### PAGING ACTIVITY

<table>
<thead>
<tr>
<th>MVS/ESA</th>
<th>SYSTEM ID MVS1</th>
<th>DATE 01/01/96</th>
<th>INTERVAL 30.00.00</th>
<th>RPT VERSION 5.2.0</th>
<th>TIME 12.00.00</th>
<th>CYCLE 1.000 SECONDS</th>
</tr>
</thead>
</table>

**OPT = IEAOPT42**

### CENTRAL STORAGE PAGING RATES - IN PAGES PER SECOND

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>SWAP BLOCK</th>
<th>NON TOTAL RATE</th>
<th>SWAP TOTAL RATE</th>
<th>SUM TOTAL RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEABLE SYSTEM AREAS (NON VIO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPA</td>
<td>0.10</td>
<td>0.10</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>CSA</td>
<td>0.00</td>
<td>0.00</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>SUM</td>
<td>0.10</td>
<td>0.10</td>
<td>100</td>
<td>0.00</td>
</tr>
</tbody>
</table>

| ADDRESS SPACES | | | | |
| HIPERSPACE | 0.00 | 0.00 | 0 | 0.00 | 0.00 | 0 |
| VIO | 0.00 | 0.00 | 0 | 1.25 | 1.25 | 100 |
| NON VIO | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| SUM | 0.00 | 0.00 | 0.00 | 0.00 | 1.25 | 1.25 |

| TOTAL SYSTEM | | | | |
| HIPERSPACE | 0.00 | 0.00 | 0 | 0.00 | 0.00 | 0 |
| VIO | 0.00 | 0.00 | 0 | 1.25 | 1.25 | 100 |
| NON VIO | 0.00 | 0.00 | 0.10 | 0.10 | 100 | 0.00 | 0.00 | 0 |
| SUM | 0.00 | 0.00 | 0.10 | 0.10 | 100 | 0.00 | 1.25 | 1.25 |

| SHARED | | | | |
| | 0.02 | 0.02 | | | | 0.05 | 0.05 |

**PAGE MOVEMENT WITHIN CENTRAL STORAGE** | 2.22 | **PAGE MOVEMENT TIME %** | 0.0 |
**AVERAGE NUMBER OF PAGES PER BLOCK** | 0.0 |
**BLOCKS PER SECOND** | 0.00 |
**PAGE-IN EVENTS (PAGE FAULT RATE)** | 0.10 |

---

*Figure 14. RMF Monitor I Paging Report (Central Storage)*
### Paging Activity

**MVS/ESA**

**System ID**: MVS1  
**Date**: 04/20/96  
**Interval**: 30.00.000  
**RPT Version**: 5.2.0  
**Time**: 12.00.00  
**Cycle report class**: SECONDS

**OPT = IEAOPT42**

#### Expanded Storage Movement Rates - in Pages Per Second

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Pages</th>
<th>HIPERSPACE</th>
<th>Pages</th>
<th>VIO</th>
<th>Pages</th>
<th>Shared</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Written to</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp Store</td>
<td>349.04</td>
<td>0.11</td>
<td>348.56</td>
<td>0.00</td>
<td>0.00</td>
<td>0.34</td>
<td>0.72</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Read from</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp Store</td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Freed</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without MIGRATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp Store</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>MIGRATED</td>
<td>254</td>
<td>2,893</td>
<td>0</td>
<td>416</td>
<td>4</td>
<td>1,286</td>
<td>416</td>
<td>4</td>
</tr>
<tr>
<td><strong>TOTAL STORAGE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>12.00.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Frame and Slot Counts

<table>
<thead>
<tr>
<th></th>
<th>Central Storage</th>
<th>Expanded Storage</th>
<th>Local Page Data Set Slot Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Available</strong></td>
<td>Min 81,565</td>
<td>Max 93,067</td>
<td>Avg 87,870</td>
</tr>
<tr>
<td><strong>SQA</strong></td>
<td>Min 5,712</td>
<td>Max 5,798</td>
<td>Avg 5,762</td>
</tr>
<tr>
<td><strong>LPA</strong></td>
<td>Min 1,481</td>
<td>Max 1,617</td>
<td>Avg 1,561</td>
</tr>
<tr>
<td><strong>CSA</strong></td>
<td>Min 1,669</td>
<td>Max 1,797</td>
<td>Avg 1,732</td>
</tr>
<tr>
<td><strong>LSQA</strong></td>
<td>Min 3,540</td>
<td>Max 5,201</td>
<td>Avg 3,985</td>
</tr>
<tr>
<td>REGIONS+SWA</td>
<td>Min 801,729</td>
<td>Max 1,310</td>
<td>Avg 253,331</td>
</tr>
<tr>
<td><strong>Total Frames</strong></td>
<td>Min 129,248</td>
<td>Max 129,248</td>
<td>Avg 129,248</td>
</tr>
<tr>
<td><strong>Fixed Frames</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nucleus</strong></td>
<td>Min 3,004</td>
<td>Max 3,004</td>
<td>Avg 3,004</td>
</tr>
<tr>
<td><strong>SQA</strong></td>
<td>Min 5,548</td>
<td>Max 5,634</td>
<td>Avg 5,598</td>
</tr>
<tr>
<td><strong>LPA</strong></td>
<td>Min 165</td>
<td>Max 170</td>
<td>Avg 168</td>
</tr>
<tr>
<td><strong>CSA</strong></td>
<td>Min 1,058</td>
<td>Max 1,058</td>
<td>Avg 1,058</td>
</tr>
<tr>
<td><strong>LSQA</strong></td>
<td>Min 2,694</td>
<td>Max 4,310</td>
<td>Avg 3,099</td>
</tr>
<tr>
<td>REGIONS+SWA</td>
<td>Min 402</td>
<td>Max 1,777</td>
<td>Avg 793</td>
</tr>
<tr>
<td>BELOW 16 M</td>
<td>Min 209</td>
<td>Max 245</td>
<td>Avg 217</td>
</tr>
<tr>
<td><strong>Total Frames</strong></td>
<td>Min 12,871</td>
<td>Max 15,467</td>
<td>Avg 13,632</td>
</tr>
</tbody>
</table>

**Total Frames**

- Central Storage: 11,200 frames
- Expanded Storage: 1,200 frames
- Fixed Total: 800 frames
- Below 16 M: 100 frames
- Auxiliary Slots: 200 frames
- Total: 14,600 frames
### Paging Activity

**MVS/ESA**

<table>
<thead>
<tr>
<th>SYSTEM ID</th>
<th>DATE</th>
<th>INTERVAL</th>
<th>CYCLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS1</td>
<td>03/14/96</td>
<td>14.59.946</td>
<td>1.000 SECONDS</td>
</tr>
</tbody>
</table>

**RPT VERSION**

<table>
<thead>
<tr>
<th>TIME</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16.30.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OPT = IEAOPT40**

<table>
<thead>
<tr>
<th>S W A P P L A C E M E N T A C T I V I T Y</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>----- AUX STORAGE ------</em></td>
</tr>
</tbody>
</table>

**AUX STOR**

| TERMINAL | CT | 14,279 | 0 | 0 | 0 | 14,279 | 13,069 | 0 | 410 | 0 | 410 | 14,272 |
| INPUT/OUTPUT | RT | 15.87 | 0.00 | 0.00 | 0.00 | 15.87 | 15.41 | 0.00 | 0.46 | 0.00 | 0.46 | 15.84 |
| W A I T | % | 97.7% | 0.0% | 0.0% | 0.0% | 100.0% | 97.1% | 0.0% | 2.9% | 0.0% | 2.9% | 100.0% |

| LONG | CT | 122 | 0 | 0 | 0 | 122 | 109 | 0 | 13 | 0 | 13 | 123 |
| W A I T | % | 0.8% | 0.0% | 0.0% | 0.0% | 100.0% | 89.3% | 0.0% | 10.7% | 0.0% | 10.7% | 100.0% |

| DETECTED | CT | 176 | 0 | 0 | 0 | 176 | 134 | 0 | 42 | 0 | 42 | 175 |
| W A I T | % | 1.2% | 0.0% | 0.0% | 0.0% | 100.0% | 76.1% | 0.0% | 23.9% | 0.0% | 23.9% | 100.0% |

| UNILATERAL | CT | 32 | 0 | 0 | 0 | 32 | 30 | 0 | 2 | 0 | 2 | 33 |
| W A I T | % | 0.2% | 0.0% | 0.0% | 0.0% | 100.0% | 93.8% | 0.0% | 6.2% | 0.0% | 6.2% | 100.0% |

**TOTAL**

| CT | 14,699 | 0 | 0 | 0 | 14,699 | 14,142 | 0 | 467 | 0 | 467 | 14,603 |
| RT | 16.23 | 0.00 | 0.00 | 0.00 | 16.23 | 15.71 | 0.00 | 0.52 | 0.00 | 0.52 | 16.24 |
| % | 100.0% | 0.0% | 0.0% | 0.0% | 100.0% | 96.8% | 0.0% | 3.2% | 0.0% | 3.2% | 100.0% |

**AUXILIARY STORAGE - AVERAGE PAGES PER SWAP OUT -**

| 0 | AVERAGE PAGES PER SWAP IN - | 0 |

**OCCURRENCES OF TERMINAL OUTPUT WAIT -**

| 550 |

---

**Figure 16. RMF Monitor I Paging Report (Swap Placement)**
3.3.3.2 User-Defined Fixed Storage

Part of the common (LPA/CSA) and private area fixed page frames can be attributed to installation-tailored requests. In the case of an IMS MVS system, there are two optional ways to specify user-defined fixed storage:

- **MVS**

  Modules in the pageable link pack area can be fixed at IPL. Restrict the fixed modules to those that are referenced frequently, as the modules remain fixed until the next IPL. Only the IMS and RMF resource cleanup modules need to be in the MVS fix list for the MVS/ESA system.

  Specify only those modules in the IEAFIXxx member of SYS1.PARMLIB. If any LPA modules still require fixing, and the pageable copy resides below 16 MB, move them to SYS1.LINKLIB to free the virtual storage below 16 MB.

  The modules that are fixed in MVS by means of the IEAFIXxx member are included in the MVS nucleus frame count.

- **IMS/ESA**

  Some virtual storage in IMS is always fixed by IMS initialization. The amount of storage varies depending on the size of the IMS system being executed. The following is a list of the storage that is always fixed:

  - **IMS Log Buffers**

    The size of the area that is page fixed is the log buffer size times the number of buffers specified, rounded up to a page boundary.

  - **IMS Interregion Communication Dispatching Blocks**

    The size of the area that is page fixed is approximately
    
    \[ 4 \text{ KB} + (600 \times \text{number of dependent regions}) \]
    
  - **IMS Miscellaneous**

    This comprises the storage management control blocks (DFSZIBnn) and the OSAM I/O driver channel end appendage (DFSAOS70). These amount to about 12 KB.

  - **Fast Path Buffers**

    For a Fast Path buffer to be used, it must first be page fixed. Assume that all buffers can be page fixed, and use the DBBF value or issue a /DIS POOL FPDB command. Adding the AVAIL and UNFIXED values yields the total number of buffers. Multiply this by the buffer size, then add an extra 25% for buffer prefixes and other fixed control information.

  - **Other IMS Fixed Storage**

    The number of pages fixed depends on the IMS system being executed. The size of the fixed storage can be found on the output from the IMS /DIS POOL ALL command (Figure 31 on page 90), under the heading of SP # and value of 228. Total the CURR value to determine the amount of storage that is fixed.

    IMS storage that can be optionally fixed by installation specification consists of modules, blocks, and pools in the control region private area, DLS address-space private area, and CSA. A percentage of the IMS nucleus in the control region address space can also be page fixed, but it is not recommended. The IMS fix list member, DFSFIxxx in IMSESA.PROCLIB,
contains entries designating the installation fix list requirements for IMS. This storage is fixed for the duration of time that the IMS online system is in operation.

The size of the fixed modules can be obtained from the IMS System Definition link-edit output, and the size of the fixed blocks is available in the output from the /DIS POOL ALL command.

An alternative to page fixing is the use of storage isolation. Storage isolation can be implemented for the control region, DL/I address space, or CSA (or a combination).

Storage isolation protects the working set size (that is, the pages being referenced) rather than fixing a set of pages, some of which may never be referenced. This reduces the number of real page frames that IMS is required to hold for the duration of time it is in the system.

Refer to 3.3.3.3, “System Paging” and 8.9, “Storage Isolation” on page 158 for a detailed description of the use of storage isolation for an IMS system.

- Modules That Are Always Fixed

The following modules are always fixed by IMS in CSA and account for about 48 KB of fixed storage.7

- DFSFXC30
- DFSREP00
- DFSXMSRB
- DFSIOSBG

3.3.3.3 System Paging

Paging consists of demand paging, virtual I/O paging, and swap paging. In systems that use TSO, swap paging almost always occurs. Demand paging and non-TSO swap paging are the areas that can be addressed to improve overall system performance. Therefore, it is necessary to have a good understanding of the paging that occurs in the system being evaluated.

Appendix C worksheet on page 382 is used to document the information about system paging.

**Total System Paging Rate:** The total system paging rate, from the RMF paging activity report page 1 (Figure 14 on page 52), is the sum of the total system, total rate for page ins plus the sum of the total system, total rate for page outs.

**Total Demand Paging Rate:** The total demand paging rate, from the RMF paging activity report, is the sum of three values: namely the total system non-VIO for nonswap/block for page in, the total system non-VIO for nonswap/nonblock for page in, and the total system non-VIO for nonswap for page out.

**Total Swap Paging Rate:** The number of total system swap page ins added to the number of swap page outs yields the swap paging rate.

The number of page movements within central storage specifies the rate of total

---

7 The exact size of these modules can be determined from the CDE entries under the LLE entry of an IMS dump or from the directory entry of IMSESARESLIB.
page movements between storage locations above 16 MB and storage locations less space below 16 MB.

The RMF Monitor I paging activity report averages the page faults over the specified interval. In certain cases, the average paging rate does not reflect the true interference caused by the high peaks in the system paging rates as reflected in the RMF Monitor II SPAG report (Figure 17). Scanning the paging columns labeled LPA IN, CSA IN, SWP OUT, PGS-SWPD IN and OUT, PRIV IN and PRV OUT, and V&H I+O, select a few samples that look high and calculate a paging burst.

In particular, use the SPAG report to check for bursts of paging in the IMS service address spaces at system checkpoint times.

<table>
<thead>
<tr>
<th>TIME</th>
<th>LPA</th>
<th>CSA</th>
<th>SWP</th>
<th>PGS-SWPD</th>
<th>PRIV_IN</th>
<th>PRV</th>
<th>V&amp;H</th>
<th>TAR</th>
<th>HI</th>
<th>ES</th>
<th>MIG</th>
<th>ESF</th>
<th>MIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:06:45</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>0</td>
<td>251</td>
<td>255</td>
<td>----</td>
<td>5.5</td>
</tr>
<tr>
<td>14:06:48</td>
<td>0.0</td>
<td>0.0</td>
<td>3.3</td>
<td>0.0</td>
<td>0.0</td>
<td>4.6</td>
<td>7.7</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>184</td>
<td>255</td>
<td>470</td>
</tr>
<tr>
<td>14:06:50</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>18</td>
<td>23</td>
<td>3.0</td>
<td>26</td>
<td>0</td>
<td>264</td>
<td>255</td>
<td>148</td>
</tr>
<tr>
<td>14:06:51</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>246</td>
<td>255</td>
<td>240</td>
</tr>
<tr>
<td>14:06:51</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>0</td>
<td>234</td>
<td>255</td>
<td>5.5</td>
<td>93</td>
</tr>
<tr>
<td>14:06:52</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>207</td>
</tr>
<tr>
<td>14:06:53</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>14:06:54</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>14:06:55</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Swapping: The RMF Monitor I paging activity report (Figure 16 on page 54) shows the swapping activity in the system.

The report lists the swap-out categories, the number of occurrences in each category, and the total number of swaps occurring during the RMF reporting interval. The sample shows a count of 14,609 at a rate of 16.23 per second. The total, together with the two below, is entered on the Appendix C worksheet (page 382).

If all TSO work is assigned to unique SRM performance groups, the workload activity report can determine whether TSO or batch work is causing the swapping. Looking at the sample report in Figure 18 on page 58, you see that the required value of 437, for all performance groups (PG), is in the column headed TRANSACTIONS, on the row marked #SWAPS. Subtracting this from the total swaps yields the number of swaps for other address spaces.
## WORKLOAD ACTIVITY

<table>
<thead>
<tr>
<th>TIME</th>
<th>PGN PGP DMN SLICE INTERVAL SERVICE</th>
<th>AVERAGE ABSORPTION, AVG TRX SERV RATE, TCB+SRB SECONDS, %</th>
<th>PAGE-IN RATES</th>
<th>STORAGE TRANSACTIONS AVG TRANS. TIME, STD. DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSYS = TSO TRXCLASS =</td>
<td>ACCTINFO = NO USERID =</td>
<td>TRXNAME = SRVCLASS =</td>
<td>CPU 51237 TRX SERV 23,688 BLOCK 0.00</td>
<td>MPL 0.02</td>
</tr>
<tr>
<td>0002 1 003 ** IOC 4,325 ABSPRTN 25,956 SINGLE 0.00</td>
<td>AVERAGE 1,132.7 AVG 0.02 TRX 000.00.00.020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCB 5.4 SHARED 0.00</td>
<td>TOTAL 25.61 ENDED 320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRB 6,258</td>
<td>HSP 0.00</td>
<td>CENTRAL 25.61 END/SEC 0.43 QUE 000.00.00.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOT 431.2K RCT 0.0 HSP MISS 0.00</td>
<td>EXPAND 0.00 #SWAPS 320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PER SEC 5B6</td>
<td>IIT 0.0 EXP SNGL 0.00 SHARED 0.00</td>
<td>TOT 000.00.00.020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HST 0.0 EXP BLK 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPL% 0.8 EXP SHR 0.00</td>
<td>EXP VEL% 66.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002 2 004 ** IOC 6,355 ABSPRTN 28,296 SINGLE 0.00</td>
<td>AVERAGE 1,097.7 AVG 0.01 TRX 000.00.00.158</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU 48158 TRX SERV 28,296 BLOCK 0.00</td>
<td>MPL 0.01 SD 000.00.00.109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCB 5.1 SHARED 0.00</td>
<td>TOTAL 21.87 ENDED 56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRB 1,897</td>
<td>HSP 0.00</td>
<td>CENTRAL 21.87 END/SEC 0.07 QUE 000.00.00.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOT 416.2K RCT 0.0 HSP MISS 0.00</td>
<td>EXPAND 0.00 #SWAPS 56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PER SEC 563</td>
<td>IIT 0.0 EXP SNGL 0.00 SHARED 0.00</td>
<td>TOT 000.00.00.158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HST 0.0 EXP BLK 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPL% 0.7 EXP SHR 0.00</td>
<td>EXP VEL% 25.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002 ALL ALL ALL IOC 82650 ABSPRTN 25,110 SINGLE 0.00</td>
<td>AVERAGE 1,599.7 AVG 0.23 TRX 000.00.00.395</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU 460.0K TRX SERV 24,879 BLOCK 0.00</td>
<td>MPL 0.23 SD 000.00.01.691</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCB 48.8 SHARED 0.00</td>
<td>TOTAL 372.55 ENDED 437</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRB 26084</td>
<td>HSP 0.00</td>
<td>CENTRAL 372.55 END/SEC 0.59 QUE 000.00.00.019</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOT 4316K RCT 0.0 HSP MISS 0.00</td>
<td>EXPAND 0.33 #SWAPS 437</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PER SEC 5,847</td>
<td>IIT 0.0 EXP SNGL 0.00 SHARED 0.00</td>
<td>TOT 000.00.00.415</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HST 0.0 EXP BLK 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPL% 6.9 EXP SHR 0.00</td>
<td>EXP VEL% 66.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 18. RMF Monitor I --- Work Load Activity Report (Part I By Groups)
**WORKLOAD ACTIVITY**

<table>
<thead>
<tr>
<th>MVS/ESA</th>
<th>SYSPLEX BBHULB</th>
<th>DATE</th>
<th>INTERVAL</th>
<th>MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP5.2.0</td>
<td>RPT VERSION 5.2.0</td>
<td>04/15/96</td>
<td>15.00.000</td>
<td>GOAL</td>
</tr>
</tbody>
</table>

POLICY ACTIVATION DATE/TIME = 04/14/96 23.23.09

REPORT BY: POLICY=HOLIDAY WORKLOAD=IMSSTC SERVICE CLASS=IMSMP

ALL IMS TELLER TRANSACTIONS

---

**-TRANSACTIONS-**

<table>
<thead>
<tr>
<th>AVG</th>
<th>0.00</th>
<th>TRANSACTION TIME</th>
<th>HHH.MM.SS.TTT</th>
<th>INTERVAL</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPL</td>
<td>0.00</td>
<td>CPU</td>
<td>624.8K</td>
<td>TRX SERV</td>
<td>6,325</td>
</tr>
<tr>
<td>ENDED</td>
<td>0</td>
<td>EXECUTION</td>
<td>0.00</td>
<td>MSO</td>
<td>2684K</td>
</tr>
<tr>
<td>END/SEC</td>
<td>0.00</td>
<td>STANDARD DEVIATION</td>
<td>0.00</td>
<td>SRB</td>
<td>0.0</td>
</tr>
<tr>
<td>#SWAPS</td>
<td>0</td>
<td>TOT</td>
<td>3503K</td>
<td>RCT</td>
<td>0.1</td>
</tr>
<tr>
<td>EXECUTD</td>
<td>0</td>
<td>PER SEC</td>
<td>1,945</td>
<td>IIT</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EXP SNGL</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HST</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EXP BLK</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>APPL%</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EXP SHR</td>
<td>0.00</td>
</tr>
</tbody>
</table>

---

**SERVICE CLASSES BEING SERVED**

SAVINGS  BONDS  INSURANCE

---

*Figure 19. RMF Monitor I --- Workload Activity Report (Part II By Policy)*
Page Faults: RMF provides indicators as to which address spaces are incurring page faults. The RMF Monitor II ARD report (Figure 20) indicates the page faults for both the system area (CSA and LPA) and the private storage associated with each address space. The number reported is always in delta mode, even though the report is run in total mode. The total/delta mode is active for the TCB TIME field.

The page faults displayed in the RMF Monitor II ARD report (Figure 20) are categorized as follows:

- **LPA RT** - Page-in rate for the Link Pack Area.
- **CSA RT** - Page-in and page-out rate for the Common System Area.
- **NVI RT** - The sum of private non-VIO pages, non-swap page-ins and page-outs.

This example uses a highly tuned system that is focused on avoiding paging activity: the values illustrate the success of this effort.

Add the monitored values to the Appendix C worksheet (page 382).
Chapter 4. Monitoring the IMS Subsystem

This chapter concentrates on the IMS system itself. Section 4.1 describes the IMS monitoring tools recommended in Chapter 1.

The rest of this chapter discusses the use of these tools. Because all IMS users have the IMS monitor, this is the tool upon which most emphasis is placed. This chapter can be used in two ways:

- To obtain the key performance parameters from the monitoring reports. It explains the significance of these values and describes worksheets that are used to record the reduced monitoring data.
- To become familiar with and understand the various IMS monitor reports. It discusses more values than just those used to complete the worksheets.

4.1 IMS Monitoring Tools

Although important measurement tools are integrated into the MVS system, your installation needs additional tools to collect all of the data required for performance evaluation. A number of programs are available that provide detailed systemwide and job-related data on activities such as paging, main storage use, and CPU utilization.

The program offerings most commonly used in IMS MVS performance evaluation are the only ones discussed. Many more program offerings are available, and your installation can evaluate their usefulness. Table 8 on page 62 lists the IMS tools that are required to perform an IMS MVS tuning exercise. Measurement tools for an IMS TM system consist of the IMS monitor, IMSPARS, IMSASAP II, DBT HD Tuning Aid, and IMS utilities that process the IMS log. Some of the tools are integrated in the IMS product, and others are program offerings. These tools give a detailed picture of how IMS uses system resources. They can identify bottlenecks within IMS and, when used in conjunction with the MVS data, they show how MVS performance problems affect overall IMS performance.

The measurement tools discussed below do not provide all the data necessary for a complete evaluation of current system performance. They do not provide information about how and under what conditions each resource is used, nor do they provide information about the existing system configuration while the data is being collected. Such data, used in conjunction with the data produced by the measurement tools, provides the basic information that you must have to conduct a system performance evaluation. It is therefore extremely important to use as many techniques as possible to get information about the system. Additional sources that are available and might be used to monitor system activity are discussed in 1.8, “Performance Monitoring Information Sources” on page 17.
### Table 8 (Page 1 of 2). IMS Data Collection Routines

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Description</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS Monitor</td>
<td>IMS component</td>
<td>Traces IMS activities at the ITASK level of detail. An offline program generates reports describing IMS online characteristics. It is the primary tool for IMS tuning.</td>
<td><em>A User's Guide to the IMS Monitor (GG66-3134)</em></td>
</tr>
<tr>
<td>IMS Monitor Summary and System Analysis Program II (IMSASAP II)</td>
<td>Program Offering 5798-CHJ</td>
<td>Processes IMS monitor data to provide enhanced summary and detailed reports on system and program performance.</td>
<td><em>IMSASAP II Description Operations (SB21-1793)</em></td>
</tr>
<tr>
<td>Log Transaction Analysis Program (DFSilTA0)</td>
<td>IMS component</td>
<td>Reports data about individual transactions based on records on the IMS log.</td>
<td><em>IMS 5.1 Utilities Reference Manual: System (SC26-8035)</em></td>
</tr>
<tr>
<td>File Select and Formatting Print utility (DFSerA10)</td>
<td>IMS component</td>
<td>Formats and prints selected records from the IMS Log</td>
<td><em>IMS 5.1 Utilities Reference Manual: System (SC26-8035)</em></td>
</tr>
<tr>
<td>IMS Performance Analysis and Reporting System (IMSPARS)</td>
<td>Program Offering 5798-CQP</td>
<td>Performs statistical analysis of IMS Log and provides several reports of the analyses.</td>
<td><em>IMSPARS Description and Operations (SB21-2140)</em></td>
</tr>
<tr>
<td>IMS DBT HD Tuning Aid</td>
<td>IMS DBT Program Product 5685-093 component</td>
<td>Provides a map of how root segments are stored in the database and can assist in deciding when to reorganize the database or to change its randomization parameters.</td>
<td><em>IMS DBT HD Tuning Aid User’s Guide (SH21-0543)</em></td>
</tr>
<tr>
<td>IMS Space Monitor</td>
<td>IMS DBT Program Product 5685-093 component</td>
<td>Assist users to forecast potential space utilization problem of IMS full-function database data sets that HD pointer checker supports, and OS data sets (including VSAM data sets).</td>
<td><em>IMS DBT Space Monitor User’s Guide (SH21-0546)</em></td>
</tr>
</tbody>
</table>
Your installation can evaluate the use of the other available tools such as PI trace, PSB trace, and Fast Path trace.

### 4.1.1 IMS Monitor

The IMS monitor, integrated into the IMS product, collects information on most dispatchable events in the IMS system, except Fast Path. It is the primary tool for tuning the IMS system, analyzing application programs, validating database design, and monitoring system performance. It is also a means of inferring the effect on IMS of the interaction of the hardware and software components of the total system. The IMS monitor collects its data while the online IMS system is in operation. The monitor can be started and stopped by means of IMS commands from the IMS master terminal.

The IMS monitor output can be either a tape or a DASD data set, which can be allocated through JCL or dynamic allocation. The use of DASD with dynamic allocation is recommended. If a DASD monitor is allocated via JCL, specify DISP=SHR so that the IMS monitor reports can be generated as each Monitor run is completed. DASD output also eliminates the requirement of having a tape drive allocated to IMS. Additionally, when a tape is allocated, the allocation task is suspended until the tape has been mounted and the tape label is verified and accepted. This can take milliseconds or minutes, depending on how closely the tape operators monitor the tape allocation and mount messages.

It is essential to provide appropriate DCB information, whether in the dynamic allocation member or in the JCL. For tape, the suggested block size is 32 KB and for DASD, half-track blocking is recommended. The default number of buffers is insufficient. Refer to 9.21, “IMS Monitor Buffer Pool” on page 203 for further information.

The IMS monitor report program (DFSUTR20) is a batch program that prints summary reports and distribution details of the data collected by the IMS monitor. These reports contain buffer pool statistics: accumulated values such as elapsed time, IWAIT time, CPU time, NOT-IWAIT time, schedule-end-to-first-DL/I-call time, elapsed execution time, and queue statistics. All of these terms are explained in the glossary; an understanding of each is essential for effective use of the IMS monitor reports.

An IMS system profile for full-function transactions can be obtained from the IMS monitor reports. A system profile consists of the number of transactions processed per second, number of transactions processed per schedule, number of DL/I calls

---

8 There used to be potential operational problems if using dynamic allocation for the DASD monitor data set. These problems were removed in IMS V5 with APAR PN78394
per transaction, number of IWAITs per transaction, CPU cycles per transaction, and region occupancy. Familiarity with profiles can be valuable in capacity planning for new applications and projected increases in the IMS workload.

The IMS monitor reports to be printed can be selected at execution time by means of the analysis control data set, included in the IMS monitor report program JCL. The time required for the processing and the printing of reports and the storage required to produce the reports depend on which reports are requested and the duration of the online trace. The basic summary reports are printed automatically. There are two optional reports: the DL/I call summary report and the distribution appendix.

Always request the DL/I call summary report. Request the distribution appendix only when you need a detailed analysis of the distribution of IWAIT times, elapsed times, and so on. The distribution parameters can be changed by specifying additional control cards to the IMS monitor report program.

4.1.2 Log Transaction Analysis Utility (DFSILTA0)

The IMS log transaction analysis utility collects information about individual transactions from the IMS system log. It generates a report that identifies each transaction, the class and priority of the transaction, and the length of time the transaction is in the system. It then calculates the total transaction response time. The definition of response time in this report is the sum of the time on the input queue, the processing time, and the time on the output queue. This information enables an installation analyst to isolate problem areas in the IMS system. It can be used to quickly identify problem transactions and to evaluate or aid in setting up class-scheduling algorithms.

Execution-time parameters include the specification of a start time and the subsequent number of checkpoints to be included from the log. To correspond with the IMS monitor data, the report can be run to process only the portion of the IMS log that was produced at the same time. The program can also produce another IMS log that includes only a subset of the input log records, which might expedite some subsequent analysis.

When DFSILTA0 is run as a batch job, execution time and CPU utilization are high, and the report can be long. Therefore, subsetting should be done to limit the use of system resources. If wait-for-input (WFI) transactions have been processed with a high processing-limit count (greater than 1000), DFSILTA0 can execute for a very long time and occupy a large amount of virtual and real storage as it builds in-storage tables for all transactions that are processed at a single scheduling.

4.1.3 IMS File Select and Formatting Print Utility (DFSERA10)

The file select and formatting print utility can be used with IMS and its related databases. Its primary function is to assist in the examination and display of data from the IMS log data set. The utility can:

- Print or copy an entire log data set.
- Print or copy from multiple log data sets based on control statement input.
- Select and print log records on the basis of sequential position in the data set.
- Select and print external trace data sets.
Select and print log records based on data contained within the record itself, such as the contents of a time, date, or identification field.

Allow exit routines to perform special processing of selected log records. For example, DFSERA30 is an exit that formats deadlock trace records.

The file select and formatting print utility uses a series of control statements to define the input and output options, selection ranges, and various field and record selection criteria.

4.1.4 IMS Performance Analysis and Reporting System (IMSPARS)

IMSPARS is a performance and analysis program that processes data from an IMS log. This program executes under the program offering called system for generalized performance analysis reporting and produces reports in graphical and tabular format. It is a valuable aid for the performance analysis and tuning of an IMS system. It produces a set of reports that can be distributed to both management and system programmers in order for them to gain a deeper understanding of the performance of an IMS system.

On many occasions, a performance problem is reported some time after the event, and may well have occurred at a time when the IMS monitor was not switched on. In this case, log analysis using products such as IMSPARS is the only way to address the problem.

IMSPARS addresses the following areas of log analysis:

- **Transaction transit-time analysis report**
  This report presents times for four components of the transaction transit time: input queue time, processing time, output queue time, and program switch time. The report can be summarized on different criteria depending on input parameters.

- **IMS resource usage (IRU)**
  This report provides statistics on the use of most of the IMS pools and resources.

- **Message queue utilization report**
  This report shows message queue data-set utilization for the LGMSG and SHMSG queues and indicates inefficient message-queue record lengths.

- **Database activity report**
  This report shows the number of blocks updated and the number of each type of update for each database.

- **Database trace report**
  This report shows the before and after versions of each changed portion of a segment and, for each change, identifies the program, transaction, region, and time. This report also shows the database block or CI size and the access method of the data sets.

- **Region histograms**
  This report is a graphical display of region activity that shows the times a region is active or idle. The report also shows the transactions scheduled, checkpoint start and stop times, and idle time for the regions.
• DC queue-manager trace report
  This report shows a time-sequenced list of each TM event such as input, message enqueue, get unique, output, free DRRN. It also presents the message content.

• CPU usage reports
  These reports show statistics of the CPU times and elapsed times for transactions, regions, and programs within the time period of the report. These reports let you determine which transaction or program is using excessive CPU time compared with other transactions or programs.

Some of the IMSPARS reports are further described in Appendix A, “Use of IMSPARS in Performance Analysis” on page 347.

4.1.5 IMS Log Data

The IMS log contains other data that can be used in performance evaluation. This data can be printed by using the log formatting print utility (DFSERA10), IMSPARS, or a user-written program that selects and formats the records of interest. The other data and associated log records are:

• X'37x3' — Lockmax
• X'4001' — Control region CPU time
• X'4001' — DLISAS address space CPU time
• X'45xx' — IMS pool statistics
  – X'4501' — Dynamic database log statistics
  – X'4502' — Queue statistics
  – X'4503' — Format buffer pool statistics
  – X'4504' — Database buffer pool statistics
  – X'4505' — Variable pool statistics
  – X'4506' — Scheduler statistics
  – X'4507' — Logger statistics
  – X'4508' — VSAM subpool statistics
  – X'4509' — Program isolation statistics
  – X'450A' — Latch statistics
  – X'450B' — CTL TCB dynamic SPA statistics
  – X'45C' — Storage pool statistics
  – X'450D' — RECANY statistics
  – X'450E' — Fixed pool statistics
  – X'450F' — Dispatcher statistics
  – X'4521' — IRLM
  – X'4522' — IRLM
• X'67FA' — LOCK trace
• X'67FE' — Program Isolation trace
• X'67FF' — Application program user abends User abends U777 (deadlock) and U778 (ROLL call), for example, can be selected and reported by analyzing these records.
4.1.6 IMS Monitor Summary and System Analysis Program II (IMSASAP II)

IMSASAP II is a performance analysis and tuning aid for IMS database and data communications systems. It processes data collected by the IMS Monitor to provide multiple reports that assist in the analysis of an IMS environment.

Some of the IMSASAP II reports are described in Appendix B, “Use of IMSASAP II” on page 361. We strongly recommend using this program offering in parallel with the IMS monitor because it calculates many of the indicators that have to be calculated manually if the IMS monitor reports are used. IMSASAP II also lists many of the reports in a sorted sequence by database name and PSB name.

4.1.7 IMS DBT HD Tuning Aid

The HD tuning aid component of IMS DBT produces the following reports that describe the distribution of root segments in HDAM and HIDAM databases:

- Actual roots per block report — Prints the actual number of roots that are stored in each database block.
- Assigned Roots per Block report — Prints the number of roots that randomize to each block.
- Assigned roots per RAP report — Prints the number of root that randomize to each root anchor point (RAP).

Typical uses of the HD tuning aid include the following kinds of performance and tuning analysis:

- Evaluation of current randomizer performance in HDAM databases
- Evaluation of current root segment locations in HIDAM databases
- Evaluation of potential randomizer performance in HDAM databases
- Evaluation of potential randomizer performance in HIDAM databases that could be converted to HDAM

Your installation's randomizing routine may have its own restrictions that can affect the way the HD tuning aid must be run. It must, of course, be capable of processing the key values that are provided as input. If the randomizer references IMS control blocks, the HD tuning aid must be executed in a batch message processing (BMP) region.

Refer to the IMS DBT HD Tuning Aid User’s Guide (SH21-0543) for a more detailed explanation of this utility.

4.1.8 IMS DBT Space Monitor

The space monitor examines DASD VTOCs and VSAM catalog entries of user-specified data sets to monitor the space utilization of IMS full-function database data sets, OS data sets or both. The collected data is logged into a data set as the space management information and is used to produce a graph report and various other types of reports. The reports include:

- Space analysis by data set report
- Summary of data sets by volume report
- Total DASD utilization by volume/device-type report
- Space monitor exception report
The space monitor assists the users to forecast potential space utilization problems of IMS full-function database data sets and OS data sets (including VSAM data sets). Refer to the IMS DBT Space Monitor User's Guide (SH21-0546) for a more detailed explanation of this utility.

4.1.9 IMS DEDB Pointer Checker Utility
This utility uses either an image copy data set or the VSAM cluster to verify the integrity of all IMS pointer values, free-space element chains, VSAM control fields, and space utilization in IMS DEDBs. It provides the comprehensive analysis information required to effectively manage the space utilization, performance characteristics, and physical attributes of IMS DEDBs. The following areas of integrity are addressed:

- All the following pointer values are verified to ensure that they refer to valid segments:
  - Root anchor point (RAP)
  - Physical twin forward (PTF)
  - Physical child first (PCF)
  - Physical child last (PCL)
  - Subset (SSP)
  - Sequential dependent (SDEP).

- All orphan segments, invalid pointer values, and broken pointer chains are detected and reported. PCF/PCL/SSP pointer interdependence errors are also detected and reported.

- All free space element (FSE) chains are checked for validity.

- Space utilization is verified by totalling the lengths of all segments, FSEs, and scraps within a CI.

- The VSAM CIDF and RDF fields are also verified to ensure that their values are correct.

Refer to the IMS DBT DEDB Pointer Checker User's Guide (SH21-0542) for a more detailed explanation of this utility.

4.1.10 IMS DEDB Tuning Aid Utility
DEDB tuning aid is an extension of the DBT DEDB pointer checker. It allows the user to evaluate the potential benefits (or impact) of changes to an existing DEDB without unloading and reloading the database. It provides a comprehensive tuning facility for those directly involved in database management, maintenance, and performance tuning. It can help the user to:

- Eliminate the guesswork in database parameter and attribute selection
- Identify and select the physical database attributes that meet the desired performance and space utilization requirements
- Simplify the evaluation and selection of a suitable randomizing routine
- Reduce maintenance frequency and costs
- Increase the productivity of database support personnel
- Evaluate the potential benefits or impact of database reorganization
- Select the normal buffer allocation (NBA) that is optimal for database reloading.

Refer to the IMS DBT DEDB Tuning Aid User's Guide (SH21-0583) for a more detailed explanation of this utility.

### 4.2 Reduction and Analysis of the IMS Monitoring Data

Once all of the measurements are complete and the reports are generated, the data must be analyzed to find the key indicators that determine the performance problems, if any, within the system. This section shows which data to consider and how to interpret the data in the performance evaluation.

#### 4.2.1 Overview of Reduction Process and Worksheets

The IMS monitor provides the data used to develop IMS system-transaction profiles, message-region workload indicators, I/O IWAIT time, and pool-space utilization indicators. Some calculations are necessary to derive the data. The sections that follow explain where to locate the data in the IMS monitor reports and how to perform the necessary calculations. If the IMSASAP II program offering is available in your installation, use it to reduce the number of manual calculations. Refer to Appendix B, “Use of IMSASAP II” on page 361 for a description of the reports that can be used.

Worksheets designed for collecting the monitored data described in 4.2.2, “IMS System Transaction Profile” can be found in Appendix C: Figure 131 on page 384 and Figure 132 on page 385.

#### 4.2.2 IMS System Transaction Profile

The IMS system transaction profile indicates the average type of transaction processed by the IMS system. The numbers are on a monitoring report, or derived from a report by dividing totals by either the number of transactions processed or the number of seconds in the trace interval.

**Trace Elapsed Time:** The trace interval is found on the run profile report (Figure 21 on page 71), next to the heading “Trace Elapsed Time in Seconds.” This is the number to be used in any “per second” calculation. In the example, the number is 177.8 seconds.

**Number of Transactions Dequeued:** The total number of transactions dequeued is found next to the heading “Total Number of Messages Dequeued” on the run profile report (Figure 21 on page 71). In the example, the number is 25,025. This number includes conversational, nonconversational, and other messages dequeued by dependent regions — in other words, all of the transaction-oriented messages dequeued during the IMS monitoring.

**Number of Transactions per Second:** The number of transactions per second is found next to the heading “Number of Transactions per Second” on the run profile report (Figure 21 on page 71). In the example, the number is 140.5.

**Number of Transactions Processed per Schedule:** The number of transactions processed per schedule is calculated by the IMS monitor and is found on the “Totals” line on the program summary report (Figure 22 on page 73), under the
heading “Trans. Deq.” This number includes any messages dequeued by a batch processing region.
IMS MONITOR **RUN PROFILE**
TRACE START 1995 108 11:00:02
TRACE STOP 1995 108 11:03:00 PAGE 0451

TRACE ELAPSED TIME IN SECONDS.......177.8
TOTAL NUMBER OF MESSAGES DEQUEUED...25025
TOTAL NUMBER OF SCHEDULES .........7771
NUMBER OF TRANSACTIONS PER SECOND 140.5
TOTAL NUMBER OF DL/I CALLS ISSUED...288737
NUMBER OF DL/I CALLS PER TRANSACTION 11.5
NUMBER OF SERVICE CALLS TO SUBSYSTEM DB2P ...10858
NUMBER OF SERVICE CALLS PER TRANSACTION 0.4
NUMBER OF COMMAND CALLS TO SUBSYSTEM DB2P .......0
NUMBER OF NORMAL CALLS TO SUBSYSTEM DB2P ...59305
NUMBER OF NORMAL CALLS PER TRANSACTION 2.3
NUMBER OF OSAM BUFFER POOL I/O'S. 0, 0.0 PER TRANSACTION
NUMBER OF MESSAGE QUEUE POOL I/O'S.......1826, 0.0 PER TRANSACTION
NUMBER OF FORMAT BUFFER POOL I/O'S.......273, 0.0 PER TRANSACTION
RATIO OF PROGRAM ELAPSED TO DL/I ELAPSED:
  REGION 45: 1.18
  REGION 73: 1.29
  REGION 74: 1.37
  REGION 75: 1.07
  REGION 76: 1.32
  REGION 77: 1.00
  REGION 78: 3.67
  REGION 79: 1.04
  REGION 80: 1.02
  REGION 81: 1.39
  REGION 82: 1.33
  REGION 83: 1.00
  REGION 84: 1.58
  REGION 85: 1.22
  REGION 86: 1.89
  REGION 87: 2.82
  REGION 88: 1.71
RATIO OF DL/I ELAPSED TO DL/I IWAIT:
  REGION 45: 1.21
  REGION 73: 2.01
  REGION 74: 1.83
  REGION 75: 1.17
  REGION 76: 1.81
  REGION 77: 1.81
  REGION 78: 3.31
  REGION 79: 1.12
  REGION 80: 1.29
  REGION 81: 1.78
  REGION 82: 1.62
  REGION 83: 1.01
  REGION 84: 1.90
  REGION 85: 1.21
  REGION 86: 1.09
  REGION 87: 527.18
  REGION 88: 1.73
RATIO OF PROGRAM ELAPSED TO OTHER SUBSYSTEM CALLS ELAPSED
  REGION 45: 10.47
  REGION 76: 2.82
  REGION 78: 1.75
  REGION 79: 259.67
  REGION 81: 2.58
  REGION 84: 2.15
  REGION 85: 10.25
  REGION 86: 93.10
  REGION 87: 1.11

...
Figure 21. IMS Monitor Run Profile Report
### IMS Monitor Program Summary Report

<table>
<thead>
<tr>
<th>PSBNAME</th>
<th>SCHEDS</th>
<th>DEQ.</th>
<th>CALLS</th>
<th>I/O</th>
<th>IWAIT</th>
<th>TRANS.</th>
<th>CPU</th>
<th>ELAPSED</th>
<th>SCHED.</th>
<th>ELAPSED</th>
<th>SCHED.</th>
<th>ELAPSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMX002</td>
<td>355</td>
<td>1393</td>
<td>21181</td>
<td>15.2</td>
<td>13583</td>
<td>0.6</td>
<td>3.9</td>
<td>62198</td>
<td>132191</td>
<td>4842</td>
<td>336898</td>
<td></td>
</tr>
<tr>
<td>INX001</td>
<td>1</td>
<td>0</td>
<td>153</td>
<td>-</td>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>10010</td>
<td>177750219</td>
<td>61402</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KBX053</td>
<td>19</td>
<td>21</td>
<td>369</td>
<td>17.5</td>
<td>181</td>
<td>0.4</td>
<td>1.1</td>
<td>14607</td>
<td>667698</td>
<td>2585</td>
<td>604108</td>
<td></td>
</tr>
<tr>
<td>CQX001</td>
<td>4</td>
<td>579</td>
<td>10922</td>
<td>18.8</td>
<td>2524</td>
<td>0.2</td>
<td>144.7</td>
<td>10010</td>
<td>26154791</td>
<td>10762699</td>
<td>180689</td>
<td></td>
</tr>
<tr>
<td>CQX008</td>
<td>6</td>
<td>877</td>
<td>26507</td>
<td>30.2</td>
<td>5948</td>
<td>0.2</td>
<td>146.1</td>
<td>10010</td>
<td>36012458</td>
<td>3109388</td>
<td>246379</td>
<td></td>
</tr>
<tr>
<td>INX002</td>
<td>15</td>
<td>21</td>
<td>359</td>
<td>17.0</td>
<td>178</td>
<td>0.4</td>
<td>1.4</td>
<td>14714</td>
<td>175107</td>
<td>3410</td>
<td>125076</td>
<td></td>
</tr>
<tr>
<td>OLP020</td>
<td>2</td>
<td>4621</td>
<td>33908</td>
<td>7.3</td>
<td>5032</td>
<td>0.2</td>
<td>3210.5</td>
<td>10010</td>
<td>47480290</td>
<td>115024</td>
<td>20549</td>
<td></td>
</tr>
<tr>
<td>CLX002</td>
<td>3</td>
<td>871</td>
<td>12042</td>
<td>13.8</td>
<td>4555</td>
<td>0.3</td>
<td>290.3</td>
<td>10010</td>
<td>65348718</td>
<td>251453</td>
<td>225081</td>
<td></td>
</tr>
<tr>
<td>JNX004</td>
<td>2</td>
<td>1664</td>
<td>10758</td>
<td>6.4</td>
<td>4171</td>
<td>0.3</td>
<td>832.0</td>
<td>10010</td>
<td>66154977</td>
<td>55198</td>
<td>79513</td>
<td></td>
</tr>
<tr>
<td>GLX003</td>
<td>88</td>
<td>90</td>
<td>1076</td>
<td>11.9</td>
<td>606</td>
<td>0.5</td>
<td>1.0</td>
<td>12969</td>
<td>202155</td>
<td>519</td>
<td>197663</td>
<td></td>
</tr>
<tr>
<td>SWX220</td>
<td>61</td>
<td>417</td>
<td>5553</td>
<td>13.3</td>
<td>1095</td>
<td>0.1</td>
<td>6.8</td>
<td>38124</td>
<td>868993</td>
<td>73416</td>
<td>127118</td>
<td></td>
</tr>
<tr>
<td>ASX220</td>
<td>182</td>
<td>1257</td>
<td>15875</td>
<td>12.6</td>
<td>3131</td>
<td>0.1</td>
<td>6.9</td>
<td>39605</td>
<td>701364</td>
<td>8200</td>
<td>101549</td>
<td></td>
</tr>
<tr>
<td>FXO03</td>
<td>26</td>
<td>33</td>
<td>973</td>
<td>29.4</td>
<td>637</td>
<td>0.6</td>
<td>1.2</td>
<td>99161</td>
<td>1225747</td>
<td>20176</td>
<td>965740</td>
<td></td>
</tr>
<tr>
<td>SIX002</td>
<td>7</td>
<td>9</td>
<td>92</td>
<td>10.2</td>
<td>48</td>
<td>0.5</td>
<td>1.2</td>
<td>9334</td>
<td>494327</td>
<td>493</td>
<td>384476</td>
<td></td>
</tr>
<tr>
<td>FXO05</td>
<td>94</td>
<td>105</td>
<td>2678</td>
<td>25.5</td>
<td>2099</td>
<td>0.7</td>
<td>1.1</td>
<td>17770</td>
<td>178184</td>
<td>559</td>
<td>159517</td>
<td></td>
</tr>
<tr>
<td>RXO01</td>
<td>394</td>
<td>1863</td>
<td>19946</td>
<td>10.7</td>
<td>3811</td>
<td>0.1</td>
<td>4.7</td>
<td>43453</td>
<td>447288</td>
<td>3072</td>
<td>94595</td>
<td></td>
</tr>
<tr>
<td>CIX002</td>
<td>215</td>
<td>371</td>
<td>5234</td>
<td>14.1</td>
<td>1312</td>
<td>0.2</td>
<td>1.7</td>
<td>20163</td>
<td>282862</td>
<td>4388</td>
<td>163923</td>
<td></td>
</tr>
<tr>
<td>TDX004</td>
<td>65</td>
<td>105</td>
<td>2687</td>
<td>25.5</td>
<td>1946</td>
<td>0.7</td>
<td>1.6</td>
<td>36290</td>
<td>748218</td>
<td>10444</td>
<td>463182</td>
<td></td>
</tr>
<tr>
<td>LCX000</td>
<td>95</td>
<td>221</td>
<td>18963</td>
<td>85.8</td>
<td>16845</td>
<td>0.8</td>
<td>2.3</td>
<td>115784</td>
<td>1583329</td>
<td>8447</td>
<td>680616</td>
<td></td>
</tr>
<tr>
<td>TSX004</td>
<td>23</td>
<td>26</td>
<td>1113</td>
<td>42.8</td>
<td>913</td>
<td>0.8</td>
<td>1.1</td>
<td>36622</td>
<td>1310721</td>
<td>19958</td>
<td>1159484</td>
<td></td>
</tr>
<tr>
<td>TSX001</td>
<td>41</td>
<td>58</td>
<td>2289</td>
<td>39.4</td>
<td>1871</td>
<td>0.8</td>
<td>1.4</td>
<td>48557</td>
<td>1279468</td>
<td>3422</td>
<td>904451</td>
<td></td>
</tr>
<tr>
<td>JNX001</td>
<td>1</td>
<td>1382</td>
<td>6907</td>
<td>4.9</td>
<td>22</td>
<td>0.0</td>
<td>1382.0</td>
<td>10010</td>
<td>34924594</td>
<td>45274</td>
<td>25271</td>
<td></td>
</tr>
</tbody>
</table>

**TOTALS** 7771 25025 358900 14.3 134375 0.3 3.2 19975 480691 17864 149268

*Figure 22. IMS Monitor Program Summary Report*
**Number of DL/I Calls:** The total number of DL/I calls is found next to the heading “Total Number of DL/I Calls Issued” on the run profile report (Figure 21 on page 71). The total number of DL/I calls from the IMS monitor includes the DL/I call for a primed message queue record. In the example, the number is 288,737.

**Number of DL/I Calls per Transaction:** The number of DL/I calls per transaction is found next to the heading “Number of DL/I Calls per Transaction” on the run profile report (Figure 21 on page 71). This number includes TM, DB, and CHKPT calls. In the example, the number is 11.5.

**Number of Message Queue I/Os:** The number of message queue I/Os can be found next to the heading “Number of Message Queue Pool I/O’s” on the run profile report (Figure 21 on page 71). In the example, the number is 1826. The IMS monitor also adds the number of IWAITS for purge completion and the IWAITS for a DECB. The number of these IWAITS is usually very small; therefore, the number calculated by the run profile report may be used. The IWAITS for a DECB is the waiting by the queue manager for the data-event block, which is an MVS control block associated with physical I/O to an external device.

**Number of Message Queue I/Os per Transaction:** The number of message queue I/Os per transaction is found next to the heading “Number of Message Queue Pool I/O’s per Transaction” on the run profile report (Figure 21 on page 71). This field is also found on the message queue pool statistics report. In the example, the number is 0.0.

**Number of Schedules:** The total number of schedules represents how many times during the trace interval the IMS scheduler completed the scheduling process for an application program. This number is found on the Totals line on the program summary report (Figure 22 on page 73), under the heading “No. Scheds.”

**Number of OSAM I/Os:** The IMS monitor reports OSAM I/Os by subpools in the reports headed “Data Base Buffer Pool” (see Figure 23 on page 75). For the purpose of establishing a system transaction profile, the number is the total number of I/Os required (it is divided by the total number of DL/I calls to give an average number of I/Os per call).

For each subpool, the total number of I/Os is calculated by adding together three values from the Difference column: “Number of Read I/O Requests,” “Number of Single Block Writes by Buffer Steal Routine,” and “Number of Blocks Written by Purge.”

Adding together all the totals for all the subpools yields the value to be entered on the worksheet.

---

9 As part of the scheduling process, the first message for the program is copied from the message queue to the dependent region to make it immediately available. This is called message priming, and the first message-GU call is thus very efficient. Subsequent GU calls drive sync point processing and fetch the next message off the queue. The final call gets the QC status code to terminate the program. Therefore, the number of message-GU calls is one more than the number of messages processed.
Figure 23. IMS Monitor OSAM Buffer Pool Report
If IMSASAP II is available, the System IWAIT report (Figure 124 on page 370) gives a one-line entry that totals all OSAM IWAITS issued during the monitor run. If this report is used, manual calculations are not required.

If IMSPARS is used, and the internal resource usage report is requested IMSPARS produces a report for each subpool similar to the IMS monitor report. It also produces a report showing the combined values for all subpools.

---

**Note**

If the “Number of Single Block Writes by Buffer Steal Routine” is other than zero for any subpool, and the system was not running BMPs, then the subpool is almost certainly too small. See Chapter 11, “IMS Database Access Performance” on page 237 for more details.

---

**Number of VSAM I/Os:** The total number of VSAM I/Os can be calculated from the IMS monitor VSAM buffer pool report (Figure 24 on page 77), by adding the Difference columns for the “Number of CTRL Intervals Read from External Storage,” the “Number of VSAM Writes Initiated by IMS,” and the “Number of VSAM Writes to Make Space in the Pool.” Then add all of the totals for the subpools, which yields the total system number of VSAM I/Os. To calculate the total number of I/Os for Subpool ID 1 in Figure 24 on page 77, add 9926, 3531 and 0, for a total of 13457. The three other VSAM subpools in the system are not shown in the sample report. Adding the totals for all subpool IDs gives the total number of VSAM I/Os.

If IMSASAP II is available, the system IWAIT report (Figure 124 on page 370) gives a one-line entry that totals all VSAM IWAITS issued during the monitor run. If this report is used, manual calculations are not required.

If IMSPARS is used, and the internal resource usage report is requested (see Figure 117 on page 358), it produces a report for each subpool, similar to the IMS monitor report. It also produces a report showing the combined values for all subpools.

---

**Note**

If the “Number of VSAM Writes to Make Space in the Pool” is other than zero for any subpool, and the system was not running BMPs, then the subpool is almost certainly too small. See Chapter 11, “IMS Database Access Performance” on page 237 for more details.
**IMS Monitor VSAM Buffer Pool Report**

- **FIX INDEX/BLOCK/DATA**: N/N/N
- **SHARED RESOURCE POOL ID**: P1
- **SHARED RESOURCE POOL TYPE**: D
- **SUBPOOL ID**: 1
- **SUBPOOL BUFFER SIZE**: 4096
- **NUMBER hiperspace buffers**: 5000
- **TOTAL BUFFERS IN SUBPOOL**: 1000

<table>
<thead>
<tr>
<th></th>
<th>Start Trace</th>
<th>End Trace</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER OF RETRIEVE BY RBA CALLS RECEIVED BY BUF HNDLR</td>
<td>6369404</td>
<td>6435007</td>
<td>65603</td>
</tr>
<tr>
<td>NUMBER OF RETRIEVE BY KEY CALLS</td>
<td>13695062</td>
<td>13712346</td>
<td>17284</td>
</tr>
<tr>
<td>NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS</td>
<td>36</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS</td>
<td>285416</td>
<td>286714</td>
<td>1298</td>
</tr>
<tr>
<td>NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL</td>
<td>521406</td>
<td>524003</td>
<td>2597</td>
</tr>
<tr>
<td>NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED</td>
<td>2692</td>
<td>2692</td>
<td>0</td>
</tr>
<tr>
<td>NUMBER OF SYNCHRONIZATION CALLS RECEIVED</td>
<td>223254</td>
<td>224925</td>
<td>1671</td>
</tr>
<tr>
<td>NUMBER OF WRITE ERROR BUFFERS CURRENTLY IN THE SUBPOOL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LARGEST NUMBER OF WRITE ERRORS IN THE SUBPOOL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NUMBER OF VSAM GET CALLS ISSUED</td>
<td>17813958</td>
<td>1785512</td>
<td>41554</td>
</tr>
<tr>
<td>NUMBER OF VSAM SCHBF (search buffer) CALLS ISSUED</td>
<td>185436</td>
<td>186389</td>
<td>953</td>
</tr>
<tr>
<td>NUMBER OF TIMES CTRL INTERVAL REQUESTED ALREADY IN POOL</td>
<td>17370380</td>
<td>17403752</td>
<td>33372</td>
</tr>
<tr>
<td>NUMBER OF CTRL INTERVALS READ FROM EXTERNAL STORAGE</td>
<td>1776657</td>
<td>1786583</td>
<td>9926</td>
</tr>
<tr>
<td>NUMBER OF VSAM WRITES INITIATED BY IMS</td>
<td>685096</td>
<td>688627</td>
<td>3531</td>
</tr>
<tr>
<td>NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL</td>
<td>277</td>
<td>277</td>
<td>0</td>
</tr>
<tr>
<td>NUMBER OF VSAM READS FROM HIPERSPACE BUFFERS</td>
<td>412277</td>
<td>416398</td>
<td>4121</td>
</tr>
<tr>
<td>NUMBER OF VSAM WRITES TO HIPERSPACE BUFFERS</td>
<td>2190536</td>
<td>2204615</td>
<td>14079</td>
</tr>
<tr>
<td>NUMBER OF FAILED VSAM READS FROM HIPERSPACE BUFFERS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NUMBER OF FAILED VSAM WRITES TO HIPERSPACE BUFFERS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**QUOTIENT**: \( \frac{\text{TOTAL NUMBER OF VSAM READS + VSAM WRITES}}{\text{TOTAL NUMBER OF TRANSACTIONS}} = 0.53 \)

*Figure 24. IMS Monitor VSAM Buffer Pool Report*
If the background write (BGWRT) option (refer to 2.1, “IMS Control Region Execution Parameters” on page 23 for a description) is specified in the DFSVSMxx member in IMSESA.PROCLIB, there can be a discrepancy between the total number of VSAM I/Os reported in the buffer pool statistics for VSAM and the IWAITs for VSAM captured by the IMS monitor. This discrepancy occurs because the control region or the DLS address space performs the background write, and IMS cannot charge the I/O to a PST. There is no problem in this area, but it is mentioned here because you may try to add all of the VSAM IWAITs shown on the IMS Monitor and correlate them back to the VSAM buffer pool statistics.

By adding the OSAM I/Os to the VSAM I/Os and dividing the results by the number of transactions dequeued, the total number of database I/Os per transaction can be determined. This may not be an accurate statistic if batch message processes that do not dequeue any messages are executing. The I/Os for the BMPs accumulate in the total I/Os for the IMS system and tend to inflate the number of I/Os per transaction. Eliminate I/Os charged to BMPs do not dequeue messages by subtracting the number of I/O IWAITs for BMPs from the Total IWAITs on the program summary report (Figure 22 on page 73). The difference is the number of I/O IWAITs that the online transactions issued, which is a more accurate profile of the IMS online transactions.

**Number of Database I/Os per DL/I Call:** The total number of database I/Os is calculated by adding the number of OSAM I/Os and the number of VSAM I/Os. To calculate the number of I/Os per call, divide the sum by the number of DL/I calls.

The number that is found in the program summary report under the heading I/O IWAITs/CALL can be misleading, because it does not include the background writes. This calculated number is the number of I/Os that are synchronous with the transactions. It is unfortunate that the background writes are not included in this calculation, because they take resources that could be used by IMS or MVS. The number of these I/Os could vary on each report because they are very dependent on the mix of transactions and BMPs that are executing at the time of the monitor run. The number of background write I/Os tends to be higher when BMPs are executing.

**Number of Format I/Os:** The number of format I/Os can be found on two reports: the message format buffer pool report, and the the run profile report. Using the message format buffer pool report (Figure 32 on page 95), the sum of the “Number of Directory I/O Operations,” 6, and the “Number of I/F I/O’s,” 267, yields the total number of message format pool I/Os, 273. This sum can also be found under the heading “Number of Format Buffer Pool I/O’S” on the run profile report (Figure 21 on page 71). In the example, the number is also 273.

**Number of Format I/Os per Transaction:** The number of format I/Os per transaction can be found under the heading “Number of Format Buffer Pool I/O’s per Transaction” on the run profile report (Figure 21 on page 71). In the example, the number is 0.0. This number can also be found on the message format buffer pool report (Figure 32 on page 95).

**Number of ACBLIB I/Os:** The I/Os to the ACBLIB, if any occurred, are shown on the region IWAIT report (Figure 25 on page 80) for each region. Under the heading “Scheduling and Termination,” any occurrences of PSB=, INT=, or DMB indicate I/Os that resulted from obtaining PSBs, intent lists, or DMBs needed to complete scheduling of that program or transaction. Add each of these
occurrences for each region and then total the sums. The total is the number of ACBLIB I/Os that occurred during the IMS monitor interval.

The IMSASAP II system IWAIT report (Figure 124 on page 370) has a one-line entry that shows all the ACBLIB I/Os done.

**Number of ACBLIB I/Os per Schedule:** The total number of ACBLIB I/Os divided by the number of schedules yields the number of ACBLIB I/Os per schedule.

**Number of I/Os per Transaction:** To compute the number of I/Os per transaction, add the number of message queue I/Os, OSAM I/Os, VSAM I/Os, format I/Os, and ACBLIB I/Os. Numbers not included in this calculation are PGMLIB I/Os for programs, TP line I/Os, and IMS logging I/Os.

---

**Note**

The number of I/Os in the program summary report may not agree with the total I/Os calculated from the buffer pool statistics if the BGWRT option is active. Refer to the preceding explanation of VSAM I/Os.
<table>
<thead>
<tr>
<th><strong>REGION</strong></th>
<th>8S OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>FUNCTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLI CALLS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>60438</td>
<td>30219</td>
<td>30836</td>
<td>DD=CIDA5613</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>70560</td>
<td>35280</td>
<td>35442</td>
<td>DD=CIDA5513</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14243</td>
<td>14243</td>
<td>14243</td>
<td>DD=CIDA5622</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>39256</td>
<td>39256</td>
<td>39256</td>
<td>DD=CIDA5522</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24361</td>
<td>24361</td>
<td>24361</td>
<td>PI=TIDA58</td>
<td>...1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35234</td>
<td>35234</td>
<td>35234</td>
<td>DD=CMDD48</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>26467</td>
<td>26467</td>
<td>DD=CMDD38</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>32174</td>
<td>32174</td>
<td>DD=CMDB85PR</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>32786</td>
<td>10928</td>
<td>DD=FADC9305</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>50932</td>
<td>25466</td>
<td>DD=CIDA5604</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>71753</td>
<td>23917</td>
<td>DD=FADC9405</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>15416</td>
<td>15416</td>
<td>DD=CIDA5623</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>26241</td>
<td>13120</td>
<td>DD=CIDA5523</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>25781</td>
<td>25781</td>
<td>DD=CIDA5504</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>131323</td>
<td>65661</td>
<td>DD=PLDD2508</td>
<td>VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>448</td>
<td>14373358</td>
<td>32083</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2P CALLS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>36929</td>
<td>1605</td>
<td>4780</td>
<td>TERM THRD</td>
<td>D50</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>12776</td>
<td>532</td>
<td>1133</td>
<td>COMMIT PH.1</td>
<td>P10</td>
<td></td>
</tr>
<tr>
<td>791</td>
<td>2015683</td>
<td>2548</td>
<td>156312</td>
<td>NORMAL CALL</td>
<td>PRO</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>11203</td>
<td>487</td>
<td>610</td>
<td>CREATE THRD</td>
<td>CT0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>4973</td>
<td>207</td>
<td>1609</td>
<td>SIGNON</td>
<td>S00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>885</td>
<td>2081564</td>
<td>2352</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 25. IMS Monitor Region IWAIT Report
4.2.3 Region Performance Indicators

The region indicators show the amount of work being done in each dependent region and how efficiently the work is being processed. The Region Summary report (Figure 26 on page 83 through Figure 28 on page 85) contains the region indicators.

4.2.3.1 Scheduling and Region Occupancy

Four reports give you data on scheduling and occupancy.

Mean Schedule NOT-IWAIT Time: The schedule NOT-IWAIT time is the amount of IMS control-region or dependent-region NOT-IWAIT time used to perform scheduling for a dependent region. This time can include page fault resolution time and time waiting for the CPU. It can be differ from the elapsed time, as all IMS scheduling IWAIT time is subtracted from it. In this case, the number to be analyzed is the mean for the heading “Scheduling and Termination NOT-IWAIT Time” for the overall system. It is listed on the Totals line of the “Scheduling” section of the region summary report. If the system average is greater than 10 to 12 ms, then further investigation is needed. Using Figure 26 on page 83, the total mean NOT-IWAIT time is 2757 µs (2.7 ms), and is well within the guideline.

Schedule End to First DL/I Call: Schedule end to first DL/I call is the elapsed time between the end of scheduling and the first TM DL/I call issued by the newly scheduled program. This time can include:

- Program load
- High-level language initialization
- Application program initialization
- Loading of subroutines
- Waiting for CPU (due to higher priority tasks)
- Page faulting

Using the “Mean Elapsed Time” column in the region summary report (Figure 27 on page 84), note any regions that have a mean time greater than 300 ms. A region with only one occurrence need not be included in the analysis because it is usually a BMP or wait-for-input transaction, and the load time does not affect the response-time objectives for that region.

Even though the mean time is within the guidelines, there still may be reasons to investigate program load times. The Maximum column can be an indicator of other problems in program load. If this field has large numbers (over one second), review load times for the individual applications. There can be program load times over the guideline, and these times may be acceptable if the programs involve load modules from 750 KB to 2 MB. These load modules are candidates for VLF, virtual fetch, or program preload. The programs by region report (Figure 30 on page 87) contains the “Scheduling End to First Call” time for each application program within each region.

Program Elapsed Time: This value is not necessarily an indicator of the health of the IMS system. It may be perfectly acceptable to have some transactions that do major amounts of work take 2 seconds. On the other hand, a transaction taking 0.25 second might be performing badly if it does only one or two I/Os. Consequently, the program elapsed time is not a value that is entered on the worksheets. Instead, an assessment is made, for the most active transactions, as to what is a reasonable processing time given the number of calls and I/Os. This is
compared with the actual processing elapsed times, as discussed in 4.2.7.1, “Application Programs” on page 105.

All the same, program execution elapsed times can be used for a first-pass evaluation of a system as an indicator of general health. This technique is used in Chapter 5, “Identifying IMS Performance Problems” on page 109. The IMS monitor program summary report (Figure 22 on page 73) provides this information in the rightmost column headed “Elapsed Time/Trans.”

---

**Note**

The “Elapsed Execution” section of the IMS monitor region summary report (Figure 28 on page 85) can be misleading. It reports elapsed time per schedule (although there can be minor discrepancies when the monitor is started just after a program schedule), not elapsed time per transaction.

---

**Region Occupancy:** Region occupancy, computed by the IMS monitor (see Figure 29 on page 86), is the sum of the dependent region’s scheduling and termination time, the schedule end to first DL/I call, and the elapsed execution times, divided by the trace elapsed time. It helps to address questions such as:

- Are more dependent regions required to process the IMS workload?
- Are all the existing regions required?
- Should IMS class scheduling be reevaluated to better balance the workload?

A region occupancy of 65% is suggested as safe for throughput in a system where peak workload is expected. This value is derived from the Pollaczek-Khintchine mean value theorem:

\[
\text{Wait time} = \frac{\text{expected time in queue}}{(1 - \rho)}
\]

where \(\rho\), in our case, is region occupancy. This is based on a homogeneous transaction environment. It is designed to keep the service time of the region within reasonable tolerances. Using this formula, when a transaction requires scheduling and the region is running at 50% occupancy, the time to service the transaction ranges from its normal service time to twice its normal service time. At 66.7%, this range is between the normal service time for this transaction and three times the normal service time. As the value of region occupancy rises above 66.7%, the range continues to increase.

If the region is working at a consistent occupancy rate, a higher value is reasonable. The only exception would be dependent regions that process wait-for-input transactions, whose occupancy is 100% by definition.

Another way to calculate the region occupancy is to subtract the IWAIT time for “No Messages” in the “Scheduling and Termination” section of the region IWAIT report (Figure 25 on page 80) from the elapsed time of the monitor and divide this time by the elapsed time of the monitor.
### IMS Monitor Region Summary Report — Scheduling and Termination

<table>
<thead>
<tr>
<th><strong>SCHEDULING AND TERMINATION</strong></th>
<th><strong>OCCURRENCES</strong></th>
<th><strong>...ELAPSED TIME...</strong></th>
<th><strong>NOT IWAIT TIME(ELAPSED-IWAIT)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>MEAN</strong></td>
<td><strong>MAXIMUM</strong></td>
<td><strong>TOTAL</strong></td>
</tr>
<tr>
<td><strong>REGION 45</strong></td>
<td>169</td>
<td>514844</td>
<td>3046</td>
</tr>
<tr>
<td><strong>REGION 73</strong></td>
<td>30</td>
<td>21752</td>
<td>725</td>
</tr>
<tr>
<td><strong>REGION 74</strong></td>
<td>83</td>
<td>39290</td>
<td>473</td>
</tr>
<tr>
<td><strong>REGION 75</strong></td>
<td>173</td>
<td>124174</td>
<td>717</td>
</tr>
<tr>
<td><strong>REGION 76</strong></td>
<td>350</td>
<td>2848399</td>
<td>8138</td>
</tr>
<tr>
<td><strong>REGION 77</strong></td>
<td>1</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 78</strong></td>
<td>101</td>
<td>177855</td>
<td>1760</td>
</tr>
<tr>
<td><strong>REGION 81</strong></td>
<td>385281</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 82</strong></td>
<td>21752</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 84</strong></td>
<td>498606</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 85</strong></td>
<td>400267</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 87</strong></td>
<td>2660</td>
<td>35271</td>
<td>35271</td>
</tr>
<tr>
<td><strong>REGION 88</strong></td>
<td>183</td>
<td>453111</td>
<td>2476</td>
</tr>
<tr>
<td><strong>REGION 90</strong></td>
<td>1043</td>
<td>636609</td>
<td>1043</td>
</tr>
<tr>
<td><strong>REGION 91</strong></td>
<td>89</td>
<td>13923</td>
<td>1099</td>
</tr>
<tr>
<td><strong>REGION 92</strong></td>
<td>1327</td>
<td>113582</td>
<td>946</td>
</tr>
<tr>
<td><strong>REGION 93</strong></td>
<td>84</td>
<td>646315</td>
<td>1099</td>
</tr>
<tr>
<td><strong>REGION 94</strong></td>
<td>153</td>
<td>202450</td>
<td>1323</td>
</tr>
<tr>
<td><strong>REGION 95</strong></td>
<td>369</td>
<td>420024</td>
<td>1138</td>
</tr>
<tr>
<td><strong>REGION 96</strong></td>
<td>89</td>
<td>196568</td>
<td>2208</td>
</tr>
<tr>
<td><strong>REGION 97</strong></td>
<td>13</td>
<td>13923</td>
<td>1071</td>
</tr>
<tr>
<td><strong>REGION 98</strong></td>
<td>265263</td>
<td>1551</td>
<td>127680</td>
</tr>
<tr>
<td><strong>REGION 99</strong></td>
<td>140</td>
<td>201424</td>
<td>1438</td>
</tr>
<tr>
<td><strong>REGION 100</strong></td>
<td>64</td>
<td>2712829</td>
<td>42387</td>
</tr>
<tr>
<td><strong>REGION 101</strong></td>
<td>336</td>
<td>2828210</td>
<td>8417</td>
</tr>
<tr>
<td><strong>REGION 102</strong></td>
<td>82</td>
<td>41927</td>
<td>511</td>
</tr>
<tr>
<td><strong>REGION 103</strong></td>
<td>127</td>
<td>224934</td>
<td>1771</td>
</tr>
<tr>
<td><strong>REGION 104</strong></td>
<td>195</td>
<td>433159</td>
<td>2221</td>
</tr>
<tr>
<td><strong>REGION 105</strong></td>
<td>117</td>
<td>64775</td>
<td>553</td>
</tr>
<tr>
<td><strong>REGION 106</strong></td>
<td>29</td>
<td>21586</td>
<td>744</td>
</tr>
<tr>
<td><strong>REGION 107</strong></td>
<td>10</td>
<td>14487</td>
<td>1448</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>7705</td>
<td>2124637</td>
<td>2757</td>
</tr>
</tbody>
</table>

Figure 26. IMS Monitor Region Summary Report — Scheduling and Termination
## IMS Monitor Region Summary Report — Schedule to First Call

<table>
<thead>
<tr>
<th>SCHEDULE TO FIRST CALL</th>
<th>OCCURRENCES</th>
<th>Elapsed Time</th>
<th>Not Iwait Time (Elapsed - Iwait)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TOTAL</td>
<td>MEAN</td>
<td>MAXIMUM</td>
</tr>
<tr>
<td><strong>REGION 45</strong></td>
<td>170</td>
<td>460736</td>
<td>2710</td>
</tr>
<tr>
<td><strong>REGION 73</strong></td>
<td>31</td>
<td>128060</td>
<td>4130</td>
</tr>
<tr>
<td><strong>REGION 74</strong></td>
<td>84</td>
<td>127636</td>
<td>1519</td>
</tr>
<tr>
<td><strong>REGION 75</strong></td>
<td>174</td>
<td>725753</td>
<td>4170</td>
</tr>
<tr>
<td><strong>REGION 76</strong></td>
<td>351</td>
<td>905870</td>
<td>2580</td>
</tr>
<tr>
<td><strong>REGION 77</strong></td>
<td>1</td>
<td>63868</td>
<td>63868</td>
</tr>
<tr>
<td><strong>REGION 78</strong></td>
<td>102</td>
<td>828143</td>
<td>8119</td>
</tr>
<tr>
<td><strong>REGION 81</strong></td>
<td>369</td>
<td>1332826</td>
<td>3611</td>
</tr>
<tr>
<td><strong>REGION 82</strong></td>
<td>174</td>
<td>600810</td>
<td>3452</td>
</tr>
<tr>
<td><strong>REGION 84</strong></td>
<td>359</td>
<td>1131859</td>
<td>3152</td>
</tr>
<tr>
<td><strong>REGION 85</strong></td>
<td>163</td>
<td>469352</td>
<td>2879</td>
</tr>
<tr>
<td><strong>REGION 87</strong></td>
<td>9</td>
<td>822146</td>
<td>91349</td>
</tr>
<tr>
<td><strong>REGION 88</strong></td>
<td>184</td>
<td>696704</td>
<td>3786</td>
</tr>
<tr>
<td><strong>REGION 90</strong></td>
<td>610</td>
<td>1817594</td>
<td>2979</td>
</tr>
<tr>
<td><strong>REGION 91</strong></td>
<td>128</td>
<td>387173</td>
<td>3024</td>
</tr>
<tr>
<td><strong>REGION 92</strong></td>
<td>589</td>
<td>1651436</td>
<td>2803</td>
</tr>
<tr>
<td><strong>REGION 94</strong></td>
<td>154</td>
<td>468881</td>
<td>3044</td>
</tr>
<tr>
<td><strong>REGION 97</strong></td>
<td>370</td>
<td>1079705</td>
<td>2918</td>
</tr>
<tr>
<td><strong>REGION 99</strong></td>
<td>90</td>
<td>493160</td>
<td>5479</td>
</tr>
<tr>
<td><strong>REGION 100</strong></td>
<td>14</td>
<td>33596</td>
<td>2399</td>
</tr>
<tr>
<td><strong>REGION 101</strong></td>
<td>172</td>
<td>553870</td>
<td>3220</td>
</tr>
<tr>
<td><strong>REGION 102</strong></td>
<td>141</td>
<td>429322</td>
<td>3044</td>
</tr>
<tr>
<td><strong>REGION 103</strong></td>
<td>65</td>
<td>134182</td>
<td>2064</td>
</tr>
<tr>
<td><strong>REGION 104</strong></td>
<td>337</td>
<td>963927</td>
<td>2859</td>
</tr>
<tr>
<td><strong>REGION 108</strong></td>
<td>83</td>
<td>240529</td>
<td>2897</td>
</tr>
<tr>
<td><strong>REGION 109</strong></td>
<td>128</td>
<td>478238</td>
<td>3736</td>
</tr>
<tr>
<td><strong>REGION 110</strong></td>
<td>196</td>
<td>558247</td>
<td>2848</td>
</tr>
<tr>
<td><strong>REGION 113</strong></td>
<td>118</td>
<td>374197</td>
<td>3171</td>
</tr>
<tr>
<td><strong>REGION 114</strong></td>
<td>30</td>
<td>4350371</td>
<td>145012</td>
</tr>
<tr>
<td><strong>REGION 115</strong></td>
<td>11</td>
<td>32181</td>
<td>2925</td>
</tr>
</tbody>
</table>

**TOTALS** 7769 138822378 17868

Figure 27. IMS Monitor Region Summary Report — Schedule to First Call
### IMS Monitor Region Summary Report — Elapsed Execution

<table>
<thead>
<tr>
<th><strong>ELAPSED EXECUTION</strong></th>
<th><strong>OCCURRENCE</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
<th><strong>NOT IWAIT TIME (ELAPSED-IWAIT)</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 45</td>
<td>45</td>
<td>170</td>
<td>22488637</td>
<td>132286</td>
<td>1518790</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 73</td>
<td>73</td>
<td>31</td>
<td>27216002</td>
<td>877935</td>
<td>3047847</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 74</td>
<td>74</td>
<td>84</td>
<td>3673775</td>
<td>43735</td>
<td>265689</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 75</td>
<td>75</td>
<td>174</td>
<td>60594058</td>
<td>348241</td>
<td>3564357</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 76</td>
<td>76</td>
<td>351</td>
<td>65088391</td>
<td>185437</td>
<td>3606447</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 77</td>
<td>77</td>
<td>1</td>
<td>147131</td>
<td>147131</td>
<td>147131</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 78</td>
<td>78</td>
<td>102</td>
<td>93236519</td>
<td>914083</td>
<td>9210325</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 81</td>
<td>81</td>
<td>369</td>
<td>71477942</td>
<td>193707</td>
<td>6256654</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 82</td>
<td>82</td>
<td>174</td>
<td>19334892</td>
<td>111120</td>
<td>1341080</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 84</td>
<td>84</td>
<td>359</td>
<td>87402696</td>
<td>243461</td>
<td>9203155</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 85</td>
<td>85</td>
<td>163</td>
<td>21338878</td>
<td>130913</td>
<td>1873028</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 87</td>
<td>87</td>
<td>9</td>
<td>8937068</td>
<td>993007</td>
<td>2287712</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 88</td>
<td>88</td>
<td>184</td>
<td>67117323</td>
<td>364768</td>
<td>3144290</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 90</td>
<td>90</td>
<td>611</td>
<td>81279712</td>
<td>133027</td>
<td>3700063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 91</td>
<td>91</td>
<td>128</td>
<td>80846016</td>
<td>631609</td>
<td>3017809</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 92</td>
<td>92</td>
<td>589</td>
<td>84691106</td>
<td>143787</td>
<td>3909538</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 94</td>
<td>94</td>
<td>154</td>
<td>46304622</td>
<td>300679</td>
<td>3915709</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 97</td>
<td>97</td>
<td>370</td>
<td>92488285</td>
<td>249968</td>
<td>2999549</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 99</td>
<td>99</td>
<td>90</td>
<td>64054794</td>
<td>711719</td>
<td>5701565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 100</td>
<td>100</td>
<td>14</td>
<td>21142505</td>
<td>1510178</td>
<td>6858774</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 101</td>
<td>101</td>
<td>172</td>
<td>24486017</td>
<td>142360</td>
<td>1787469</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 102</td>
<td>102</td>
<td>141</td>
<td>45613742</td>
<td>323501</td>
<td>2756042</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 103</td>
<td>103</td>
<td>65</td>
<td>17655725</td>
<td>271626</td>
<td>2452872</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 108</td>
<td>108</td>
<td>83</td>
<td>5200905</td>
<td>62661</td>
<td>1021631</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 109</td>
<td>109</td>
<td>128</td>
<td>50420283</td>
<td>393908</td>
<td>2567713</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 110</td>
<td>110</td>
<td>196</td>
<td>60144288</td>
<td>306858</td>
<td>5906289</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 113</td>
<td>113</td>
<td>118</td>
<td>81157796</td>
<td>687777</td>
<td>5294026</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 114</td>
<td>114</td>
<td>30</td>
<td>25792587</td>
<td>859752</td>
<td>3194003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 115</td>
<td>115</td>
<td>11</td>
<td>168094486</td>
<td>1528136</td>
<td>82307322</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>7771</td>
<td>3735454681</td>
<td>480691</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 28. IMS Monitor Region Summary Report — Elapsed Execution*
### IMS Monitor Region Summary Report — Idle for Intent and Region Occupancy

<table>
<thead>
<tr>
<th>REGION OCCUPANCY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION 75</strong></td>
</tr>
<tr>
<td><strong>REGION 76</strong></td>
</tr>
<tr>
<td><strong>REGION 77</strong></td>
</tr>
<tr>
<td><strong>REGION 78</strong></td>
</tr>
<tr>
<td><strong>REGION 79</strong></td>
</tr>
<tr>
<td><strong>REGION 80</strong></td>
</tr>
<tr>
<td><strong>REGION 81</strong></td>
</tr>
<tr>
<td><strong>REGION 82</strong></td>
</tr>
<tr>
<td><strong>REGION 83</strong></td>
</tr>
<tr>
<td><strong>REGION 84</strong></td>
</tr>
<tr>
<td><strong>REGION 85</strong></td>
</tr>
<tr>
<td><strong>REGION 86</strong></td>
</tr>
<tr>
<td><strong>REGION 87</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDLE FOR INTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 75</td>
</tr>
<tr>
<td><strong>REGION</strong> 76</td>
</tr>
<tr>
<td><strong>REGION</strong> 85</td>
</tr>
<tr>
<td><strong>REGION</strong> 88</td>
</tr>
<tr>
<td><strong>REGION</strong> 90</td>
</tr>
<tr>
<td><strong>REGION</strong> 91</td>
</tr>
<tr>
<td><strong>REGION</strong> 92</td>
</tr>
<tr>
<td><strong>REGION</strong> 99</td>
</tr>
</tbody>
</table>

**TOTALS**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>18191773</td>
<td>454794</td>
</tr>
</tbody>
</table>

**CHECKPOINT**

NONE
**IMS MONITOR ****PROGRAMS BY REGION****

Trace Start 1995 108 11:00:02  
Trace Stop 1995 108 11:03:00  
Page 0244

<table>
<thead>
<tr>
<th><strong>REGION</strong></th>
<th><strong>OCCURRENCES</strong></th>
<th><strong>EXECUTION TIME</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GLX003</td>
<td>34</td>
<td>7094191</td>
<td>208652</td>
<td>1518790</td>
<td>17623</td>
<td>518</td>
<td>708</td>
<td></td>
</tr>
<tr>
<td>LNX002</td>
<td>11</td>
<td>1796891</td>
<td>163353</td>
<td>392653</td>
<td>39685</td>
<td>3607</td>
<td>11233</td>
<td></td>
</tr>
<tr>
<td>OLP200</td>
<td>43</td>
<td>2093253</td>
<td>47361</td>
<td>252612</td>
<td>196976</td>
<td>4580</td>
<td>10689</td>
<td></td>
</tr>
<tr>
<td>ASX230</td>
<td>10</td>
<td>618685</td>
<td>61868</td>
<td>553770</td>
<td>5508</td>
<td>550</td>
<td>595</td>
<td></td>
</tr>
<tr>
<td>STX002</td>
<td>9</td>
<td>2218786</td>
<td>257642</td>
<td>1481980</td>
<td>35281</td>
<td>3920</td>
<td>9602</td>
<td></td>
</tr>
<tr>
<td>SNX008</td>
<td>35</td>
<td>6275009</td>
<td>179285</td>
<td>773182</td>
<td>96392</td>
<td>2754</td>
<td>6397</td>
<td></td>
</tr>
<tr>
<td>GLX002</td>
<td>6</td>
<td>812410</td>
<td>135401</td>
<td>181163</td>
<td>3458</td>
<td>576</td>
<td>625</td>
<td></td>
</tr>
<tr>
<td>NWX005</td>
<td>7</td>
<td>478688</td>
<td>68384</td>
<td>195815</td>
<td>3830</td>
<td>547</td>
<td>575</td>
<td></td>
</tr>
<tr>
<td>CMX001</td>
<td>7</td>
<td>591925</td>
<td>84560</td>
<td>179169</td>
<td>27537</td>
<td>3933</td>
<td>9038</td>
<td></td>
</tr>
<tr>
<td>OLP070</td>
<td>4</td>
<td>68420</td>
<td>17105</td>
<td>26978</td>
<td>18654</td>
<td>4663</td>
<td>7789</td>
<td></td>
</tr>
<tr>
<td>HTX003</td>
<td>4</td>
<td>397109</td>
<td>99277</td>
<td>163876</td>
<td>15792</td>
<td>3948</td>
<td>7769</td>
<td></td>
</tr>
</tbody>
</table>

**REGION TOTALS**  
170 22488637 132286 460736 2710

****REGION** 73

<table>
<thead>
<tr>
<th><strong>REGION</strong></th>
<th><strong>OCCURRENCES</strong></th>
<th><strong>EXECUTION TIME</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SWX220</td>
<td>31</td>
<td>27216002</td>
<td>877935</td>
<td>3047847</td>
<td>128860</td>
<td>4130</td>
<td>46097</td>
<td></td>
</tr>
</tbody>
</table>

**REGION TOTALS**  
31 27216002 877935 128860 4130

****REGION** 74

<table>
<thead>
<tr>
<th><strong>REGION</strong></th>
<th><strong>OCCURRENCES</strong></th>
<th><strong>EXECUTION TIME</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SWX221</td>
<td>50</td>
<td>2467521</td>
<td>49350</td>
<td>265689</td>
<td>62028</td>
<td>1240</td>
<td>4359</td>
<td></td>
</tr>
<tr>
<td>SWX100</td>
<td>6</td>
<td>19646</td>
<td>3274</td>
<td>5459</td>
<td>3035</td>
<td>505</td>
<td>544</td>
<td></td>
</tr>
<tr>
<td>ASX221</td>
<td>28</td>
<td>1186608</td>
<td>42378</td>
<td>108813</td>
<td>62573</td>
<td>2234</td>
<td>18545</td>
<td></td>
</tr>
</tbody>
</table>

**REGION TOTALS**  
84 3673775 43735 127636 1519

Figure 30. IMS Monitor Programs by Region Report
4.2.3.2 Region IWAITs

There are three types of reports that can be of help.

**Program I/O IWAIT Times:** The greatest single factor in program elapsed time and region occupancy is IMS database I/O IWAIT times if program load times are within reason. Therefore, it is important to decrease I/O IWAIT times wherever possible, as any decrease in I/O IWAIT time decreases program elapsed time.

The region IWAIT report (Figure 25 on page 80), lists the Total, Mean, and Maximum I/O IWAIT times for all databases accessed by a region. To determine whether dependent region I/O IWAIT times are a problem, use the Mean column in the “DL/I Calls” section. For a 3390, look for values greater than 21 ms per region. Next, scan the Maximum IWAIT times for each database in each region. If any maximum times are greater than 300 ms, further evaluation is appropriate.

The region IWAIT report shows all I/Os experienced by dependent regions (except program load and paging). These are predominantly database I/Os. However, others may appear in the report (ACBLIB, message queue, and MFS library); check their I/O times using the same guidelines as above.

**Program Isolation IWAIT Times:** Program isolation (PI) can also be an important factor in program elapsed time. PI contention is reported on the region IWAIT report under the Function column as PI=segment name...segment code. The IWAIT time for PI is the time that the region was idle while waiting for another region to release the segment that was held by PI.

If the PI IWAIT times are long (> 200 ms), investigate the application programs and put plans into place to change either the program or the database.

---

**Note**

If the PI trace is active while the IMS monitor is running, the IMS monitor shows the PI IWAIT times as being extremely low. These times usually show as less than 1 ms. This occurs because the IMS monitor SLOG call is masked by the PI trace. Also, the PI trace data shows more PI contentions than the IMS monitor due to the PI storage management latch, which is not reported on the IMS monitor. The PI storage management trace record can be identified on the PI trace report (DFSDPIT0) by the ‘X’‘FF’ in the first byte of the RBA address.

**Idle for Intent:** When a region is idle and messages are waiting that could be processed in that region but are prevented by scheduling conflicts, then the region is said to be idle for intent. A full explanation of this aspect of region nonoccupancy can be found in 10.3, “Scheduling Failures” on page 213, and the sections following it.
4.2.4 IMS Buffer Pools

Table 9 shows the names of the main buffer pools according to where they are referenced.

<table>
<thead>
<tr>
<th>POOL Name</th>
<th>System Definition Name</th>
<th>Procedure Name</th>
<th>Fix List Name</th>
<th>/DISPLAY POOL Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message queue</td>
<td>MSGQUEUE</td>
<td>QBUF</td>
<td>QBUF</td>
<td>QBUF</td>
</tr>
<tr>
<td>Message format</td>
<td>Format</td>
<td>FBP</td>
<td>MFBP</td>
<td>MFP</td>
</tr>
<tr>
<td>DMB</td>
<td>DMB</td>
<td>DMB</td>
<td>DLDP</td>
<td>DMBP</td>
</tr>
<tr>
<td>PSB</td>
<td>PSB</td>
<td>PSB</td>
<td>DLMP</td>
<td>PSBP</td>
</tr>
<tr>
<td>CSAPSB</td>
<td>CSAPSB</td>
<td>CSAPSB</td>
<td>DLMP</td>
<td>PSBP</td>
</tr>
<tr>
<td>DLIPSB</td>
<td>DLIPSB</td>
<td>DLIPSB</td>
<td>DPSB</td>
<td>PSBP</td>
</tr>
<tr>
<td>PSB work pool</td>
<td>PSBW</td>
<td>PSBW</td>
<td>PSBW</td>
<td>PSBW</td>
</tr>
<tr>
<td>OSAM buffer pool</td>
<td>N/A</td>
<td>N/A</td>
<td>DFSVSMxx member</td>
<td>DBAS</td>
</tr>
<tr>
<td>VSAM buffer pool</td>
<td>N/A</td>
<td>N/A</td>
<td>DFSVSMxx member</td>
<td>DBAS</td>
</tr>
<tr>
<td>Communication line pool</td>
<td>Dynamic pool manager</td>
<td>CIOP/HIOP</td>
<td>N/A</td>
<td>CIOP/HIOP</td>
</tr>
<tr>
<td>Scratch pad pool</td>
<td>Dynamic pool manager</td>
<td>SPAP</td>
<td>N/A</td>
<td>SPAP</td>
</tr>
<tr>
<td>Main storage pool</td>
<td>Calculated at Sysgen</td>
<td>WKAP</td>
<td>WKPL</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

The default pool sizes are specified and calculated during IMS system definition. Most pool sizes can be respecified in the IMS control region EXEC parameters, the IMS PROCLIB parameter block member (DFSPBxxx), or DFSVSMxx in the case of the database pools.

The sizes of the IMS pools can greatly affect IMS performance. Pools that are too small can cause excessive I/Os or IWAITs for pool space. Pools that are too large can affect the overall system paging rate. These IWAITs or paging interrupts increase the IMS transaction response time. There are certain trade-offs to be made when determining the optimum size of a pool, as explained in Chapter 5, "Identifying IMS Performance Problems" on page 109. However, a knowledge of the specified pool sizes and the current utilization of each pool is first necessary.

The IMS command /DISPLAY POOL ALL gives an indication of how the pools are currently being used. This display (Figure 31 on page 90), used in conjunction with the IMS Monitor output, provides the data necessary for an evaluation of present IMS pool utilization.
<table>
<thead>
<tr>
<th>NAME</th>
<th>SP#</th>
<th>CURR</th>
<th>MAX</th>
<th>GETS</th>
<th>FREES</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSB</td>
<td>228</td>
<td>28K</td>
<td>28K</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>GIOB</td>
<td>228</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OSWA</td>
<td>228</td>
<td>8K</td>
<td>8K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>GONW</td>
<td>228</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PST</td>
<td>231</td>
<td>52K</td>
<td>56K</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>DPST</td>
<td>231</td>
<td>40K</td>
<td>44K</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>SAP</td>
<td>231</td>
<td>132K</td>
<td>132K</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>GOWW</td>
<td>231</td>
<td>16K</td>
<td>16K</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>LOMM</td>
<td>0</td>
<td>324K</td>
<td>324K</td>
<td>81</td>
<td>0</td>
</tr>
<tr>
<td>D1NA</td>
<td>228</td>
<td>8K</td>
<td>8K</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>DL2W</td>
<td>0</td>
<td>8K</td>
<td>8K</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>DG2W</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>QSAV</td>
<td>231</td>
<td>4K</td>
<td>16K</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>VRPL</td>
<td>231</td>
<td>16K</td>
<td>16K</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>LSAA</td>
<td>0</td>
<td>808K</td>
<td>808K</td>
<td>201</td>
<td>1</td>
</tr>
<tr>
<td>AWE</td>
<td>231</td>
<td>112K</td>
<td>112K</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>PDIR</td>
<td>231</td>
<td>36K</td>
<td>36K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DDIR</td>
<td>231</td>
<td>28K</td>
<td>28K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LCRE</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PCIB</td>
<td>0</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SIDX</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RRE</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SMB</td>
<td>231</td>
<td>62K</td>
<td>62K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BCPT</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>GESE</td>
<td>241</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EPST</td>
<td>231</td>
<td>208K</td>
<td>208K</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>IDT</td>
<td>241</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBRC</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TTAB</td>
<td>231</td>
<td>224K</td>
<td>224K</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>EZS</td>
<td>241</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FNCB</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TCBT</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CMWU</td>
<td>231</td>
<td>12K</td>
<td>12K</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>FSRB</td>
<td>228</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>KLSD</td>
<td>0</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IRLM</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STTR</td>
<td>231</td>
<td>52K</td>
<td>52K</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>BQEL</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>USMU</td>
<td>229</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RCTE</td>
<td>231</td>
<td>3K</td>
<td>3K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SLOG</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ADSC</td>
<td>228</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FPCP</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AESL</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XPST</td>
<td>231</td>
<td>16K</td>
<td>16K</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>BXQE</td>
<td>231</td>
<td>16K</td>
<td>16K</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>SRBC</td>
<td>228</td>
<td>8K</td>
<td>8K</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>MSGP</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LPST</td>
<td>251</td>
<td>8K</td>
<td>8K</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>CULE</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RPST</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>VTCB</td>
<td>251</td>
<td>18K</td>
<td>18K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RECA</td>
<td>0</td>
<td>19K</td>
<td>19K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>GSAV</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FEIB</td>
<td>231</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IEQE</td>
<td>0</td>
<td>0K</td>
<td>0K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LS6X</td>
<td>231</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>XI24</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>STAT</td>
<td>241</td>
<td>4K</td>
<td>4K</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 31 (Part 1 of 3). IMS Command /DIS POOL ALL
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQEL 241</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>TT24 231</td>
<td>24K 24K 6 0</td>
</tr>
<tr>
<td>LGWA 231</td>
<td>104K 104K 26 0</td>
</tr>
<tr>
<td>VMA 231</td>
<td>8K 8K 2 0</td>
</tr>
<tr>
<td>LGWX 0</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>LQ 251</td>
<td>104K 104K 0 0</td>
</tr>
<tr>
<td>RCNT 251</td>
<td>2K 2K 0 0</td>
</tr>
<tr>
<td>DDRE 0</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>CCB 0</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>LGND 251</td>
<td>8K 8K 2 0</td>
</tr>
<tr>
<td>USRD 251</td>
<td>4K 4K 1 0</td>
</tr>
<tr>
<td>LS24 0</td>
<td>2K 2K 1 0</td>
</tr>
<tr>
<td>GS24 231</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>CLLE 231</td>
<td>64K 64K 16 1</td>
</tr>
<tr>
<td>QMBA 0</td>
<td>8K 8K 1 0</td>
</tr>
<tr>
<td>AHDR 251</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>XMCI 241</td>
<td>180K 180K 45 1</td>
</tr>
<tr>
<td>DBPB 231</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>LG24 0</td>
<td>12K 12K 3 0</td>
</tr>
<tr>
<td>IAIFP 231</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>RACW 231</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>S24 231</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>CLE 231</td>
<td>0 436K 436K 109 0</td>
</tr>
<tr>
<td>CBLK 251</td>
<td>0K 0K 0 0</td>
</tr>
<tr>
<td>SVPG 231</td>
<td>24K 24K 6 0</td>
</tr>
<tr>
<td>SVPL 0</td>
<td>148K 148K 37 0</td>
</tr>
<tr>
<td>S0PB 0</td>
<td>0K 0K 0 0</td>
</tr>
</tbody>
</table>

**CBT POOLS**

**MESSAGE QUEUE POOL:** BFRS/SIZE 20/2112

**SEQUENTIAL BUFFERING:** STATUS = NOT INIT

**OSAM DB BUFFER POOL:** ID 004K BSIZE 4K NBUF 1000 FX=Y/Y

**OSAM DB BUFFER POOL:** ID 008K BSIZE 8K NBUF 1000 FX=Y/Y

**OSAM DB BUFFER POOL:** ID 012K BSIZE 12K NBUF 100 FX=Y/Y

---

*Figure 31 (Part 2 of 3). IMS Command /DIS POOL ALL*
Figure 31 (Part 3 of 3). IMS Command /DIS POOL ALL
The descriptions that follow indicate where to find:

- The size of the pool
- Maximum amount of pool space used
- IWAITs for pool space
- Pool space failures and the I/Os that occur when the pool is too small.

All references are to the data on the /DIS POOL ALL output (Figure 31 on page 90) unless otherwise indicated.

Entries in the storage management section of the IMS system contents directory (SCD) list the starting address and size of each pool. This information, in conjunction with the GTF page fault trace records, can be used for an IMS pool page-fault analysis.

**PSB Pools (DPSB and PSBP):** If LSO=Y is selected, only one PSB pool (PSBP) is created, and this pool resides in ECSA. If LSO=S is selected, two PSB pools are created: the PSBP pool resides in ECSA and contains the IOPCBs, alternate PCBs, and Fast Path PCBs. The DPSB pool resides in the DL/I separate address-space extended private area.

The size of the PSB Pool (Size) and the maximum amount used (High) are displayed. I/Os for PSBs are indicated in the region IWAIT reports as IWAITs for PSB=. Any requests for PSB pool space that cannot be immediately satisfied from the pool are reported on the pool-space failure report (Figure 71 on page 218) as ID DLMP (ECSA pool) or DPSB (DL/I SAS pool).

The only way to monitor these failures is with the IMS monitor. The maximum amount of storage used (HIGH) for this pool is always close to the allocated size of the pool because of the pool space management algorithms used by IMS (see 9.13.4, “PSB Pool Space Management” on page 192).

**DMB Pool (DMBP):** If the maximum amount of the DMB pool used (High) is at least 4 KB less than the size of the pool (Size), there is enough space in the pool for all active DMBs. Any time not enough space is available to bring in another DMB, a pool-space failure is reported in the pool-space failure report (Figure 71 on page 218) as ID DLDL. These space failures show up only if the transaction has to wait for another transaction to complete in order to delete a DMB. If a DMB that is inactive can be deleted, no pool-space failure occurs. I/Os for DMBs are listed in the region IWAIT report as IWAITs for DMB=.

If there are any IWAITs for DMBs, investigate them to see whether the DMB is requestor because this is the first time it is being referenced after an IMS startup or after a /DBR command, or because the DMB was rejected when the pool was too small. If the DMB was rejected because the pool was too small, then the performance of the transaction is significantly degraded. The reason for this degradation is that IMS closes the databases referenced by the DMB that is already in the pool and then loads the new DMB and opens the databases referenced by the transaction. Also, it is likely that the DMB that was forced out of the pool will be referenced a short time later, will have to be reloaded, and the cycle starts again.

The IMS log can be used to show any open or close activity; these events build X’20’ and X’21’ log records, respectively. If IMSPARS is used, the end-of-file record counts show the counts of all the different log records.
**Message Format Pool:** The size of the format pool (Size) is the amount of storage allocated to the pool. Space is the amount of storage available for loading format blocks. The difference between Size and Space is the amount used for DCBs and statistical counters for MFS. The amount of free storage (Free) is space minus the amount currently allocated for loaded blocks. Format pool I/Os can be located on the message format buffer pool report (Figure 32 on page 95). If a request for a format cannot be met because there is not enough space in the pool, the least recently used format blocks in the pool are deleted, and the new block is fetched. An IWAIT is indicated in the communication IWAIT report (Figure 33 on page 96) as I/O=DIR=format name or I/O=BLK=format name (directory and block I/Os, respectively).

If an MFS I/O is for a MID or MOD, the Midname or Modname is shown. If it is for a DIF or DOF,\(^\text{10}\) the 6-byte device format name is prefixed with two hex characters (possibly unprintable), representing the device type and the device features. The first character of the name is in lower case for a DIF and upper case for a DOF.

A full description of the MFS pool, its use and its performance, can be found in 9.11, “Message Format Pool” on page 184.

Analyze the “Block Washed for FRE” field from the message format buffer pool report (Figure 32 on page 95) to determine whether blocks are being deleted from the pool because not enough fetch request elements (FREs) are specified. If this is the case, add additional FREs to the number already specified, and specify this new number on the IMS control region procedure using the FRE= parameter. Each FRE occupies 48 bytes.

If the free space never gets below a size that other formats can use, the possibility exists that not enough FREs are specified. The number of FREs must be large enough to fill the entire pool. Generate a list of the format library to determine the average size of the active formats in order to allocate the correct number of FREs.

**Message Queue Pool:** To find the size of the message queue pool, multiply the number of main storage queue buffers (BFRS) by the size of one queue buffer (Size). The total number of I/Os is indicated by the I/O entry on the /DIS POOL output or can be found on the run profile report (Figure 21 on page 71).

**OSAM Database Buffer Pool:** The amount of storage allocated to all of the OSAM buffer subpools can be calculated by adding the product of the subpool size and the number of buffers. The total number of I/Os can also be found in the run profile report, (Figure 21 on page 71) or by adding the size of the subpools (see Figure 23 on page 75). The buffer pool is equal to the sum of the subpools.

\(^{10}\) MID stands for message input device, and MOD stands for message output device in IMS MFS terminology.
### IMS Monitor

**Message Format Buffer Pool**

<table>
<thead>
<tr>
<th>Description</th>
<th>Start Trace</th>
<th>End Trace</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of P/F Requests</td>
<td>1775801</td>
<td>1789462</td>
<td>13661</td>
</tr>
<tr>
<td>Number of I/F Requests</td>
<td>3559468</td>
<td>3591825</td>
<td>32357</td>
</tr>
<tr>
<td>Number of I/F I/O'S</td>
<td>12317</td>
<td>12584</td>
<td>267</td>
</tr>
<tr>
<td>Number of Times Pool Compress Would Be Successful</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Directory I/O Operations</td>
<td>919</td>
<td>925</td>
<td>6</td>
</tr>
<tr>
<td>Number of Times Block Washed For Fre</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Times P/F Request Ignored</td>
<td>27058</td>
<td>27104</td>
<td>46</td>
</tr>
<tr>
<td>Number of F/B Requests</td>
<td>3559361</td>
<td>3591717</td>
<td>32356</td>
</tr>
<tr>
<td>Number of Times F/B Request Ignored</td>
<td>12</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>Number of Times I/F On F/B Queue</td>
<td>3448240</td>
<td>3478464</td>
<td>30224</td>
</tr>
<tr>
<td>Number of Times I/F On I/F Queue</td>
<td>98908</td>
<td>100774</td>
<td>1866</td>
</tr>
<tr>
<td>Number of Times F/B On I/F Queue</td>
<td>3559349</td>
<td>3591704</td>
<td>32355</td>
</tr>
<tr>
<td>Number of Times P/F On I/F Queue</td>
<td>195453</td>
<td>198558</td>
<td>3105</td>
</tr>
<tr>
<td>Number of Times P/F On F/B Queue</td>
<td>1553291</td>
<td>1563801</td>
<td>10510</td>
</tr>
<tr>
<td>Number of Times There Was No Dir Entr For A Block</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Number of Times I/O Errors Point Or Read Macro</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Immed I/O Requests Waited Due To Max I/O</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Requests Satisfied By Index/Dynamic Dir</td>
<td>11401</td>
<td>11662</td>
<td>261</td>
</tr>
</tbody>
</table>

**Quotient**: \[ \frac{\text{Total Number of Immediate Fetch I/O's} + \text{Directory I/O's Operations}}{\text{Total Number of Transactions}} \] = 0.01

---

**Figure 32.** Message Format Buffer Pool Report
<table>
<thead>
<tr>
<th>NODE OR</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>FUNCTION</th>
<th>BLKSIZE</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBC5C026</td>
<td>3</td>
<td>11523</td>
<td>3841</td>
<td>4037</td>
<td>I/O=BLK= &quot;XSS1WD&quot;</td>
<td>3133</td>
<td>MFS</td>
</tr>
<tr>
<td>SBC4A019</td>
<td>1</td>
<td>25475</td>
<td>25475</td>
<td>25475</td>
<td>I/O=BLK= &quot;FCE60&quot;</td>
<td>1384</td>
<td>MFS</td>
</tr>
<tr>
<td>SBC38018</td>
<td>5</td>
<td>298310</td>
<td>59662</td>
<td>59662</td>
<td>I/O=BLK= &quot;RWF715F&quot;</td>
<td>23</td>
<td>MFS</td>
</tr>
<tr>
<td>SBC78496</td>
<td>1</td>
<td>2542</td>
<td>2542</td>
<td>2542</td>
<td>I/O=BLK= &quot;RWF715&quot;</td>
<td>1526</td>
<td>MFS</td>
</tr>
</tbody>
</table>

Figure 33. IMS Monitor Communications IWAIT Report
**VSAM Database Buffer Pool**: If VSAM databases are specified in the system, an entry is made for each VSAM subpool. The size of each subpool can be determined by multiplying the number of buffers in the subpool (NMBUFS) by the size of each buffer (BSIZE). The total size of the buffer pool is equal to the sum of the subpool sizes and is shown in an entry (BSIZE) following all of the subpool entries. I/O operations are also indicated, but the VSAM Buffer Pool report (Figure 24 on page 77) provides the same information in greater detail.

**Scratch Pad Area Pool**: The SPAP was formerly called the communications work pool (CWAP). It is now used for one purpose only: to hold the scratch pad areas of all active and held conversations. It is one of the DFSPOOL-managed pools that can expand and contract according to demand. A control region execution parameter (SPAP=) can specify an absolute maximum size, although we normally recommend letting it default to 2 GB – 1. If the maximum value is reached, a requestor of more space waits, and an IWAIT for MVS storage group (STG), is noted. POOL=CWAP is reported in the IMS monitor region IWAIT report (Figure 25 on page 80) or the communication IWAIT report (Figure 33 on page 96).

**PSB Work Pool (PSBW)**: All work areas required for the PSBs are allocated from this pool. No I/O ever occurs for this pool. The size of the pool is listed as Size and the maximum amount of space used is listed as High. Any requests for space that cannot be satisfied from this pool are reported in the IMS monitor Reports report (Figure 71 on page 218).

**Database Work Area Pool (DBWP)**: The size of this pool is reported as Size and the maximum amount of space is reported under High. Any time a request for space cannot be satisfied, the dependent region making the request receives a pseudo-abend of U477, U801, or U804. This pool is used as a work area for segments being deleted and by other calls.

A reasonable size for this pool can be calculated using 2 KB for each PST to be allocated by the control region. After the IMS system is up and executing for a while, the high-water mark for this pool can be used to determine the new size. Allocate an additional 4 KB above this size.

The highest observed size used in this pool has been 26 KB at one installation. Most installations monitored have shown a maximum mark of 12 KB.

**Work Area Pool**: The total size is specified by Size and the maximum amount of space used by High. IWaits for STG. POOL=TPPL in the region IWAIT report indicate that requests for pool space could not immediately be satisfied. This pool is used as the /ERE command work area, the simple checkpoint work area, and IMS working storage.

The default value is only 5 KB; however, it must be specified as 20 KB or more, even though it may seem to never need this much. The demands by /ERE are unpredictable. Override the default using the control region execution parameter WKAP=. 
**Dynamic Pool Manager:** The dynamic pool manager (DFSPOOL) manages space in the following dynamic buffer pools:

- CIOP — communication input output pool (This is the only dynamic pool that exists below the 16 MB line.)
- HIOP — high input output pool
- SPAP — scratch pad area pool (formerly CWAP)
- CESS — communication external subsystem
- AOIP — automated operator (type 2) message pool
- EMHB — expedited message handler buffer
- FPWP — Fast Path work pool
- LUMP — LU 6.2 manager private buffer pool
- LUMC — LU 6.2 manager common buffer pool

**Note**

Some documentation refers to these dynamically managed pools as *fixed pools*, which may seem confusing. This is because they contain sets of fixed-size buffers. In particular, IMSPARS produces an excellent report on each pool entitled “Fixed Pool Usage Statistics.”

Each pool is defined to contain up to 32 subpools, where a subpool is a set of buffers all of the same size. More typically, just eight buffer sizes are specified. For each buffer size, the pool space is defined with a primary and a secondary allocation. The storage manager obtains storage in blocks of these sizes. The primary block can be allocated at initialization or, alternatively, at the first time a particular buffer size is needed to satisfy a request. Secondary blocks are obtained whenever more buffers of that size are needed. Secondary storage is released when no longer required. Storage allocated at initialization is never released.

More detailed information can be found in 9.2, “Dynamic Pool Management” on page 162.

**Communications Input Output Buffer Pool (CIOP/HIOP and RECANY):** These pools contain all the TP buffers for the terminals and the working storage for message editing to take place.

The CIOP provides a work area for IMS command processing. It is also used for BTAM input and output messages. If a terminal is in Test MFS mode, then MFS control blocks (MID/MOD, DIF/DOF) are read into the CIOP.

RECANY buffers receive all input from VTAM. However, input messages are quickly moved into the HIOP for editing (MFS, ISC, or basic edit), and from there to the message queue pool or Fast-Path EMH buffer pool as appropriate. All VTAM output messages use the HIOP, whereas MFS editing takes place only if appropriate. The VTAM SEND is done directly from the HIOP.

CIOP and HIOP (but not RECANY) buffers are managed by the dynamic pool manager, DFSPOOL. If a request for space cannot be satisfied, the pool is dynamically increased (see 9.2, “Dynamic Pool Management” on page 162).
4.2.5 Communication Line Reports

There are three kinds of communication line reports you can take advantage of:

**Communications Summary Report:** The communications summary report (Figure 34 on page 100) reports the time needed for the communications routine to transfer data to and from the terminals. The report shows the BTAM line number and the VTAM node name.

Note that IMS does not IWAIT on communications I/Os. One instance of a communications ITASK terminates after sending a reply message; another instance of the communications ITASK is initiated to process the SNA acknowledgment. The IMS monitor measures ITASK and IWAIT durations. Consequently, the monitor reports do not indicate the lengths of terminal I/Os.

On input from a VTAM terminal, each RECANY buffer has an associated ITASK, which quickly transfers control (IXCTL) to a communications ITASK for input message handling through the HIOP and onto the queue. Therefore, no direct correlation between transactions and communications ITASKs exists. Indeed, when MFS paging is used, there may be several times more communication ITASKs than transactions.

The IMS monitor communications reports contain information about communications ITASKs.

**Line Function Report:** The line functions report (Figure 35 on page 101) shows the size of the input and output messages that IMS processes. The entries on this report are either the BTAM line numbers or the VTAM node names.

This report can be useful in determining which lines or nodes are active and, by the size of the messages transmitted, whether the control units to which these active terminals belong could be a bottleneck.

The value for the “Turn Around Intervals Mean Interval” column are the same as for “Mean Elapsed Time” on the communication summary. In other words, the heading “Turn Around Interval” is misleading; the values represent the average elapsed time of the communications IWAITs for that terminal.

Looking at both reports together (Figure 34 on page 100 and Figure 35 on page 101), and picking the third node reported, SBC3B018, you can see that nine occurrences of the communications ITASK were used to process two inputs using RECANY buffers (Receive Blocks) and three output Sends (Trans. Blocks). The average ITASK elapsed time was 19.125 ms, of which 16.958 ms was NOT-IWAIT Time. In other words, on average, the IWAIT time was 2.167 ms over nine events, or 19.503 ms IWAIT time in total. Understanding the IWAIT time requires the communication IWAIT report.
**IMS MONITOR COMMUNICATION SUMMARY**

<table>
<thead>
<tr>
<th>NODE OR LINE NUMBER</th>
<th>OCCURRENCES</th>
<th>........ELAPSED TIME........</th>
<th>NOT IWAIT TIME (ELAPSED-IWAIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TOTAL</td>
<td>MEAN</td>
</tr>
<tr>
<td>SBC23925</td>
<td>62</td>
<td>714618</td>
<td>11526</td>
</tr>
<tr>
<td>SBC23787</td>
<td>77</td>
<td>1041740</td>
<td>13529</td>
</tr>
<tr>
<td>SBC38018</td>
<td>9</td>
<td>172132</td>
<td>19125</td>
</tr>
<tr>
<td>SBL10P04</td>
<td>71</td>
<td>1093609</td>
<td>15402</td>
</tr>
<tr>
<td>SBC23218</td>
<td>98</td>
<td>1935042</td>
<td>19745</td>
</tr>
<tr>
<td>SBC25487</td>
<td>24</td>
<td>696695</td>
<td>29028</td>
</tr>
<tr>
<td>SBC22909</td>
<td>54</td>
<td>760249</td>
<td>14078</td>
</tr>
<tr>
<td>SBC47961</td>
<td>96</td>
<td>974908</td>
<td>10155</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>35931</td>
<td>280</td>
</tr>
<tr>
<td>SBC24493</td>
<td>39</td>
<td>397657</td>
<td>10196</td>
</tr>
<tr>
<td>SBC22927</td>
<td>114</td>
<td>1441200</td>
<td>12642</td>
</tr>
<tr>
<td>SBC40547</td>
<td>8</td>
<td>113912</td>
<td>14239</td>
</tr>
<tr>
<td>SBC48911</td>
<td>19</td>
<td>152649</td>
<td>8034</td>
</tr>
<tr>
<td>SBC49239</td>
<td>49</td>
<td>571168</td>
<td>11656</td>
</tr>
<tr>
<td>SBC26125</td>
<td>41</td>
<td>572631</td>
<td>13966</td>
</tr>
<tr>
<td>SBC26577</td>
<td>116</td>
<td>2097592</td>
<td>18082</td>
</tr>
<tr>
<td>SBC23735</td>
<td>78</td>
<td>837344</td>
<td>10735</td>
</tr>
<tr>
<td>SBC24183</td>
<td>42</td>
<td>708811</td>
<td>16876</td>
</tr>
<tr>
<td>SBC49675</td>
<td>84</td>
<td>871147</td>
<td>10370</td>
</tr>
<tr>
<td>SBC26864</td>
<td>63</td>
<td>480794</td>
<td>7631</td>
</tr>
<tr>
<td>SBC22981</td>
<td>39</td>
<td>1040482</td>
<td>26679</td>
</tr>
</tbody>
</table>

*Figure 34. IMS Monitor Communication Summary*
### Figure 35. IMS Monitor Line Functions

<table>
<thead>
<tr>
<th>NODE OR DEVICE</th>
<th>LINE NUMBER</th>
<th>DEVICE TYPE</th>
<th>MEAN BLOCKS</th>
<th>MEAN BLKSIZE</th>
<th>MEAN BLKSIZE</th>
<th>MEAN INTERVALS</th>
<th>MEAN INTERVAL</th>
<th>MEAN INTERVAL</th>
<th>MEAN PAGING REQUESTS</th>
<th>MAX BLOCKS</th>
<th>MAX BLKSIZE</th>
<th>MAX BLKSIZE</th>
<th>MAX INTERVALS</th>
<th>MAX INTERVAL</th>
<th>MAX INTERVAL</th>
<th>MAX PAGING REQUESTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBC23925</td>
<td>19</td>
<td>SLU2</td>
<td>36</td>
<td>59</td>
<td>668</td>
<td>1408</td>
<td></td>
<td>11526</td>
<td>175696</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC23787</td>
<td>24</td>
<td>SLU2</td>
<td>40</td>
<td>95</td>
<td>673</td>
<td>1408</td>
<td></td>
<td>13529</td>
<td>235115</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC38018</td>
<td>2</td>
<td>SLU2</td>
<td>157</td>
<td>311</td>
<td>836</td>
<td>999</td>
<td>9</td>
<td>19125</td>
<td>148614</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBL1004</td>
<td>22</td>
<td>SLU2</td>
<td>38</td>
<td>86</td>
<td>574</td>
<td>1339</td>
<td>71</td>
<td>15402</td>
<td>285759</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC23218</td>
<td>31</td>
<td>SLU2</td>
<td>48</td>
<td>251</td>
<td>845</td>
<td>1408</td>
<td>98</td>
<td>19745</td>
<td>291497</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC25487</td>
<td>7</td>
<td>SLU2</td>
<td>34</td>
<td>59</td>
<td>654</td>
<td>1408</td>
<td>24</td>
<td>29028</td>
<td>157758</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC22909</td>
<td>17</td>
<td>SLU2</td>
<td>41</td>
<td>73</td>
<td>761</td>
<td>1408</td>
<td>54</td>
<td>14078</td>
<td>166377</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC47961</td>
<td>30</td>
<td>SLU2</td>
<td>40</td>
<td>87</td>
<td>821</td>
<td>1408</td>
<td>96</td>
<td>10155</td>
<td>129012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC SYS</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>136</td>
<td>136</td>
<td></td>
<td>128</td>
<td>280</td>
<td>1040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC24493</td>
<td>12</td>
<td>SLU2</td>
<td>40</td>
<td>68</td>
<td>680</td>
<td>1408</td>
<td>39</td>
<td>10196</td>
<td>88889</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC22927</td>
<td>35</td>
<td>SLU2</td>
<td>36</td>
<td>75</td>
<td>656</td>
<td>1408</td>
<td>114</td>
<td>12642</td>
<td>222894</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC48547</td>
<td>2</td>
<td>SLU2</td>
<td>13</td>
<td>23</td>
<td>562</td>
<td>709</td>
<td>8</td>
<td>14239</td>
<td>82170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC48911</td>
<td>5</td>
<td>SLU2</td>
<td>28</td>
<td>51</td>
<td>691</td>
<td>965</td>
<td>19</td>
<td>8034</td>
<td>83014</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC49239</td>
<td>14</td>
<td>SLU2</td>
<td>41</td>
<td>87</td>
<td>867</td>
<td>1408</td>
<td>49</td>
<td>11656</td>
<td>132562</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC26125</td>
<td>13</td>
<td>SLU2</td>
<td>41</td>
<td>79</td>
<td>786</td>
<td>1408</td>
<td>41</td>
<td>13966</td>
<td>138964</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC26577</td>
<td>36</td>
<td>SLU2</td>
<td>37</td>
<td>71</td>
<td>673</td>
<td>1408</td>
<td>116</td>
<td>18082</td>
<td>291514</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC23735</td>
<td>24</td>
<td>SLU2</td>
<td>43</td>
<td>152</td>
<td>747</td>
<td>1408</td>
<td>78</td>
<td>10735</td>
<td>157775</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC24183</td>
<td>11</td>
<td>SLU2</td>
<td>32</td>
<td>76</td>
<td>554</td>
<td>933</td>
<td>42</td>
<td>16876</td>
<td>260578</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC49675</td>
<td>23</td>
<td>SLU2</td>
<td>41</td>
<td>130</td>
<td>556</td>
<td>1408</td>
<td>84</td>
<td>10370</td>
<td>180595</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC26864</td>
<td>20</td>
<td>SLU2</td>
<td>40</td>
<td>106</td>
<td>730</td>
<td>1408</td>
<td>63</td>
<td>7631</td>
<td>115047</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC22981</td>
<td>12</td>
<td>SLU2</td>
<td>44</td>
<td>84</td>
<td>670</td>
<td>1408</td>
<td>39</td>
<td>26679</td>
<td>330644</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 4. Monitoring the IMS Subsystem
**Communication IWAIT Report:** The communication IWAIT report (Figure 33 on page 96) shows the IWAIT times for the communications ITASKs. The possible causes of these IWAITs are:

1. Non-I/O IWAITs
   - Wait for a busy dataset control block when needing to read an MFS block (report shows DCB=BLK=name)
   - Wait for a busy dataset control block when needing to read the directory entry for an MFS block (report shows DCB=DIR=name)

   The first of these is undesirable; it is a sign of very high contention for an MFS data set. The second is highly unlikely except perhaps soon after IMS startup, if the network gets very busy very quickly.

2. I/O IWAITs
   The report shows one of the following items:
   - DDNAME=SHMSG
   - DDNAME=LGMSG
   - DDNAME=QBLKS
   - I/O=DIR=format name
   - I/O=BLK=format name

   if an I/O to one of the queue data sets was necessary, or

   in the case of an MFS I/O (directory or actual block, respectively).

   If many IWAITs for I/O=BLK= are experienced, consider increasing the size of the MFS pool with the expectation of making more blocks resident. As always, this must be done with an understanding of any real storage implications. It is probably better to perform MFS I/Os than suffer high paging rates.

4.2.6 DL/I Call Summary Report
The DL/I call summary report (Figure 36 on page 104) can be used to perform detailed analysis of programs in terms of the calls issued and the IWAITs needed to satisfy these calls. The analysis then can be used to determine whether a database requires reorganization or redesign. It can also help to identify programs that are poorly written, or that have perhaps just a small error in a segment search argument (doing a sequential search of a database when the programmer thought it was using a secondary index, for example).

The report summarizes each program specification block. It shows every PCB, and for each it reports every unique combination of DL/I call function and status code.

For the most part, the report is self-explanatory. However, the following points may be useful:

- **Database Calls**
  - For each database call, the call summary reports the segment level, which is taken from the PCB. If the call is an unsuccessful get call (for example, one receiving greater or equal status), it reports the lowest level segment it could find. If it could not find even the requested root segment, then the level number is (00).
When the level number is (00) and no status code is received, it was this call that caused a deadlock, and this was the program chosen to be backed out.

- If a space write occurs because a buffer pool is too small to hold all concurrent updates, then the IWAITs for the get call include those for the space write.

- I/O PCB Calls
  
  - A GU in parentheses represents the get of a primed message. In other words, (GU) is the first GU immediately after program scheduling. Apart from slight aberrations caused when the Monitor is switched on and off while IMS is actually processing transactions, the number of message primes generally equals the number of IOPCB GU calls that receive QC status codes.
  
  - ASRT does not represent a real DL/I call. It is possible for message processing programs to be written that do not issue a final message GU call before terminating on a QC status. The ASRT call is an artificial call at program termination time to ensure that such programs have their sync point processing attributed to program elapsed time rather than to scheduling/termination time.
  
  - IWAITs for a message GU are for the sync point database updates. However, it is important to remember that, for OSAM, each IWAIT may represent a chained I/O of multiple blocks.

Using Figure 36 on page 104 as an example, scan the column IWAITS/CALL, looking for any value over three. There are circumstances where more than three I/Os (IWAITS) can occur. An example might be the case of an insertion of a root segment into a HIDAM database with a secondary index that allows multiple keys in its sequence field. In this case, the I/Os to the secondary index data set, the overflow data set, the HIDAM index data set, the actual insertion to the database data set, the changing of the old twin forward pointer, and the changing of the old twin backward pointer could result in eight or more I/Os using VSAM. Some or all of these may be necessary and unavoidable.

Make a note of any I/Os found that cannot be explained. Then analyze the databases to determine whether they need to be reorganized; a database redesign might even be contemplated.

The report also shows the elapsed time and NOT-IWAIT time for the calls. Some of the maximum NOT-IWAIT times may seem extremely long. The following are three possible reasons for this long time:

- A checkpoint was in process when the call was issued, and the call could not be completed until the checkpoint completed.

- The CPU was overcommitted, and the dependent region could not be redispached after the call to post the event completed.

- A high paging rate caused page faults in the DL/I code or pools.

Another cause of high NOT-IWAIT time could be that the logger was busy and the record could not be logged. This is not likely, unless there is a severe DASD bottleneck on the devices on which the OLDS are resided.
### IMS Monitor CALL SUMMARY Report

**Trace Start:** 1995 10/8 11:00:02  
**Trace Stop:** 1995 10/8 11:03:00  
**Page:** 0456

<table>
<thead>
<tr>
<th>PSB Name</th>
<th>PCB Name</th>
<th>Call Func</th>
<th>Lev</th>
<th>Stat Code</th>
<th>Calls</th>
<th>Iwaits</th>
<th>Elapsed Time Mean</th>
<th>Elapsed Time Maximum</th>
<th>NOT IWAIT Time Mean</th>
<th>NOT IWAIT Time Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBX053 I/O PCB</td>
<td>INIT ( )</td>
<td>19</td>
<td>0</td>
<td>0.00</td>
<td>122</td>
<td>1028</td>
<td>122</td>
<td>1028</td>
<td>122</td>
<td>1028</td>
</tr>
<tr>
<td>(GU)</td>
<td>19</td>
<td>0</td>
<td>0.00</td>
<td>20</td>
<td>23</td>
<td>20</td>
<td>23</td>
<td>20</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>ASRT ( )</td>
<td>18</td>
<td>0</td>
<td>0.00</td>
<td>1732</td>
<td>5411</td>
<td>1732</td>
<td>5411</td>
<td>1732</td>
<td>5411</td>
<td></td>
</tr>
<tr>
<td>GU ( ) QC</td>
<td>18</td>
<td>17</td>
<td>0.94</td>
<td>389319</td>
<td>896112</td>
<td>368791</td>
<td>896112</td>
<td>368791</td>
<td>896112</td>
<td></td>
</tr>
<tr>
<td>PURG ( )</td>
<td>19</td>
<td>2</td>
<td>0.10</td>
<td>9330</td>
<td>174480</td>
<td>173</td>
<td>499</td>
<td>173</td>
<td>499</td>
<td></td>
</tr>
<tr>
<td>ISRT ( )</td>
<td>19</td>
<td>0</td>
<td>0.00</td>
<td>92</td>
<td>107</td>
<td>92</td>
<td>107</td>
<td>92</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>CHNG ( )</td>
<td>19</td>
<td>0</td>
<td>0.00</td>
<td>13104</td>
<td>209958</td>
<td>13104</td>
<td>209958</td>
<td>13104</td>
<td>209958</td>
<td></td>
</tr>
<tr>
<td>GU ( )</td>
<td>2</td>
<td>2</td>
<td>1.00</td>
<td>415353</td>
<td>541528</td>
<td>365625</td>
<td>480730</td>
<td>365625</td>
<td>480730</td>
<td></td>
</tr>
</tbody>
</table>

I/O PCB SUBTOTAL: 133 | 21 | 0.15 | 62408 | 57574

| TIDA58 REPL (01)TISA581 GU (01)TISA581 | 19 | 0 | 0.00 | 279 | 345 | 279 | 345 |

DL/I PCB SUBTOTAL: 38 | 2 | 0.05 | 981 | 308

| MSDA21 REPL (01)MSSA211 GU (01)MSSA211 | 20 | 0 | 0.00 | 19 | 31 | 19 | 31 |

DL/I PCB SUBTOTAL: 40 | 0 | 0.00 | 21 | 21

| DB2P CREATE THRD | 19 | 19 | 1.00 | 454 | 508 | 0 | 0 |
| SIGNON | 21 | 21 | 1.00 | 167 | 546 | 0 | 0 |
| TERM THRD | 18 | 18 | 1.00 | 1659 | 5347 | 0 | 0 |
| COMMIT PH.2 | 20 | 20 | 1.00 | 164213 | 519310 | 0 | 0 |
| COMMIT PH.1 | 20 | 20 | 1.00 | 164552 | 437868 | 0 | 0 |
| NORMAL CALL | 60 | 60 | 1.00 | 70003 | 527754 | 0 | 0 |

SUBSYS SUBTOTAL: 158 | 158 | 1.00 | 68465

PSB TOTAL: 369 | 181 | 0.49 | 51913 | 20786

---

**Figure 36. IMS Monitor DL/I Call Summary Report**
4.2.7 Application Program Execution and Database Usage

Significant performance improvements can be made by analyzing the design of the IMS databases and how they are used by the application programs. Changes to the databases and applications are usually long-range performance modifications but are still considered part of the overall performance evaluation. Also, understanding the interrelationship between the active application programs and databases can be helpful in making overall performance decisions for the IMS system.

Refer to Chapter 11, “IMS Database Access Performance” on page 237 for a detailed description of database analysis.

The worksheet shown in Figure 133 on page 387 provides a facility for collecting IMS monitor data for further study of application program and database resources. First record the “Trace Elapsed Time in Seconds” from the monitor run profile report (see Figure 21 on page 71). Next, record the number of active message processing regions. This need not be too precise; just avoid counting any low-occupancy regions.

Multiply these two values to get the total region elapsed time.

4.2.7.1 Application Programs

All transactions need to be reviewed, but often only a few select applications and their associated transactions account for a large percentage of the IMS workload. The percentage of the IMS workload consumed by a transaction can be viewed in terms of total DL/I calls issued, total number of transactions processed, or total I/Os incurred by the application program. All of these are reflected in the total elapsed time of the program. For some installations, one program accounts for most of the workload; for other installations, it may be several programs.

To determine which application programs account for the majority of the total workload, scan the column with the heading “Calls” on the program summary report (Figure 22 on page 73). Select the ten PSBs with the highest number of DL/I calls.

Then scan the heading of I/O IWAITS and find the ten PSBs with the highest number of IWAITs. Combine the two “top ten” lists, creating a list of slightly more than ten busiest programs.

Continuing with the program summary report, compute the total elapsed time each of these PSBs uses by multiplying the number of transactions processed (TRANS. DEQ.) by the elapsed time per transaction.

When the total elapsed time is computed for all selected PSBs, arrange them in descending sequence and select those that account for more than 10% of the total region elapsed time (if the regions were 50% occupied, these PSBs would account for 20% of the total workload). If necessary, return to select more PSBs with the next highest numbers of calls or I/Os, until the greater part of the total workload is accounted for.

After the data collection is completed, you can review the programs with the application developers and database designers to determine if the application program or database design can be improved by reducing the number of calls and
I/Os. Because these programs account for the majority of the IMS workload, any improvement to them improves overall performance.

Another area in application program design to analyze is the program elapsed time. This analysis requires additional data which can be found in the program summary report and the program I/O report.

From the program I/O report (Figure 38 on page 107), note the “Mean IWAIT” time for each program. This time is on the “PSB Total” line for each PSB on the report. If a lot of transactions were processed during the monitored period, the overall total “Mean IWAIT Time” can be substituted for the “Mean PSB IWAIT Time” in the calculations. In the sample report shown in Figure 38 on page 107, this value is 28 ms (28,242 µs).

The objective is to see how well the transaction is performing relative to the amount of DL/I work it does.

Starting with the first transaction on the program summary report (Figure 22 on page 73), calculate the estimated elapsed time for each transaction, excluding BMPs and WFls. You can also exclude any transactions that have low volumes and are clearly not significantly above-average users of IMS resources. Use the formula in Figure 37 to calculate the expected elapsed time per transaction. (It is best to use the “Mean IWAIT Time” for each program, but the overall Mean IWAIT time for all the programs can be used to facilitate the analysis.)

\[
\text{ELAPSED TIME} = \frac{(\text{IWAITS/CALL} \times \text{CALLS/TRAN}) \times \text{MEAN PGM IWAIT TIME}}{\text{CPU TIME/TRAN}} + 15\% \text{ of this calculation}
\]

Figure 37. IMS Elapsed Time

Compare this value with the “Elapsed Time/Tran” from the program summary report. Any transactions that exceed their calculated value by more than 25% are subject to detailed analysis.

### 4.2.7.2 Databases

Using the monitor print utility, you can ascertain which databases are referenced most frequently only by analyzing the region IWAIT reports (Figure 25 on page 80) for each region. First scan the “DL/I Calls Occurrence” column for each region, selecting the five or six databases with the highest number of calls. Make a note of the DDnames for each region, and calculate the system totals for these most-used databases. Add the total number of occurrences for each DDname for all the regions. Rank the total number of occurrences, previously computed for each DDname, in descending sequence, until the sum is greater than 50% of the total.

Users of IMSASAP II may find this much easier. The IMSASAP IWAIT summary report (see Figure 127 on page 374) reports the totals for each database and includes the number of I/Os, DL/I calls, IWAITs/call and percentage of total calls.

To complete the data collection, two more numbers must be calculated for each of the busiest databases, using the region IWAIT report. First determine the IWAIT total by adding together the Total IWAIT Times for the DDname, in each region report in which the name appears. Calculate the sum, then divide by I/O occurrences to get the IWAIT mean time for each DDname.
Again, using the IMSASAP IWAIT summary report makes this process very easy.

Review the accumulated information with the database analyst and designers to determine whether the number of I/Os or the IWAIT time can be reduced. If this is accomplished, the response time for the transactions in the IMS system can be improved.

Figure 38. IMS Monitor Program I/O Report
This chapter is a guide to identifying any IMS performance issues. It suggests five questions about the current performance that to identify areas for further study. The rest of the chapter explains the reasoning behind the questions. The same questions apply for Fast Path systems. The difference is that some of the numbers involved may not be available from the IMS monitor.

5.1 Overview

For each of the performance issues identified, a reference is made to another chapter where further questions and detailed explanations may be found. We hope that in this way, any bottlenecks can be identified and understood and appropriate tuning can be accomplished.

There is a direct correlation between IMS performance, real storage availability, virtual storage availability, I/O activity, and CPU utilization. A shortage of real or virtual storage causes an increase in I/O activity, which in turn causes an increase in CPU utilization. Therefore, a change to any one of these areas has some effect on the use of the other resources.

The first few times through this performance analysis, it is recommended that every section be analyzed so you become familiar with the procedure and the guideline numbers. The numbers provided for the decision-making process are general guidelines and are not to be construed as absolute or minimum values. When actual (measured) numbers are greater than the guidelines, make an attempt to explain the differences. If all numbers fall within the guidelines, you probably do not have an IMS performance problem, except possibly in extreme high-volume or high-performance situations.

Note that the sections that follow are designed to provide general reference information about aspects of IMS performance. Their use is not limited to problem identification.

5.2 IMS Key Performance Indicators

Key IMS performance indicators are used to identify significant resource overcommitments. If any one of these numbers is within the guidelines, the associated resource is probably not the cause of significant IMS performance problems.

5.2.1 IMS Key Performance Indicators Chart

Figure 39 on page 110 shows the key IMS performance indicators you need to pay attention to.
IMS Key Performance Indicators

1. Is Schedule/Termination NOT-IWAIT time greater than 4 ms?
   Yes → CPU RESOURCE 6.1

2. If there is an imbalance in Region Occupancy, some greater than 65%, while others are very low, IMS class scheduling should be reevaluated.
   Yes → PROGRAM SCHEDULING 10.2

3. Are the mean I/O IWAIT times greater than 21 ms or are there maximum I/O IWAIT times greater than 300 ms?
   Yes → I/O RESOURCE 7.1

4. Are the IMS control regions or is any MPR paging more than 2-3 pages per second or is CSA paging more than 5 pages per second?
   Yes → MAIN STORAGE RESOURCE 8.1

5. Are the mean Program Elapsed Times greater than 0.5 to 1.0 seconds depending on CPU and/or application?
   Yes → DEPENDENT RGN TUNING 10.9 "OR APPLICATION DESIGN Chapter 12

6. Implement recommended changes, monitor the system again, and start at the beginning.

Figure 39. IMS Key Performance Indicators
5.2.2 Schedule/Termination NOT-IWAIT Time

The “Schedule and Termination” elapsed time from the region summary report (Figure 26 on page 83), minus the IWAIT time, is the NOT-IWAIT time used to schedule and terminate transaction processing for a dependent region. This includes any time stolen from IMS by tasks with a higher dispatching priority. IMS issues a STCK instruction before and after each event that the IMS monitor traces. Therefore, if the IMS control region is interrupted by a higher-priority MVS task, the time IMS is kept waiting is included in the NOT-IWAIT time. NOT-IWAIT also includes the time IMS is suspended, waiting for page faults to be resolved. These page faults can occur within the IMS scheduler code, control blocks, DMB pool, or the PSB pool.

It has been observed that the NOT-IWAIT time for scheduling or termination is fairly consistent from system to system if minimal paging occurs. This number is a good indication of the adequacy of the CPU resource available to the IMS control region. A mean value of less than 4 ms (8 ms for CMOS processors) is an indication that resources are adequate for IMS control region processing.

The dispatching priority of IMS determines the priority that MVS gives to the IMS control region relative to other jobs in the system. In an environment with many jobs or subsystems operating concurrently, the CPU workload of jobs or subsystems with a higher dispatching priority than IMS is a major factor in determining how much CPU resource is allocated to IMS. If the schedule NOT-IWAIT times are higher than the recommended guidelines and higher priority tasks are running in the system, examine the system CPU resource further.

If no higher priority tasks are running in the system, a high NOT-IWAIT time is an indication of a high paging rate. Paging considerations are addressed in 8.3, “System Paging” on page 139.

5.2.3 Region Occupancy

The region occupancy percentage is the total time that a message processing region is busy, divided by the IMS monitor trace interval. The significance of region occupancy depends on the basic characteristics of the system.

In an environment with many different transaction types, a high region occupancy usually means that a scheduling problem exists because transactions are waiting for a region. The queues build up, which in turn causes an increase in transactions per schedule and response time, and may lead to message queue I/Os.

In environments with a small number of dominant transaction types, the region occupancy may approach 80% to 85% before an additional message processing region is required. This is because an application program can remain in the region longer and process multiple transactions per schedule with acceptable response times. (It replaces the time to schedule and load a program with the time spent waiting on the input queue.)

Regions with wait-for-input (WFI) transactions are 100% occupied, because the programs remain scheduled in the region until the processing limit is reached.

An imbalance in region occupancy is normally caused by class scheduling. Maximum transaction throughput and a balanced region workload can be achieved with no transaction class or priority scheduling. Class scheduling is necessary
when a particular class of transactions requires better response times at the expense of lower priority transactions in the system. Sometimes class scheduling may be necessary to avoid program isolation conflicts.

Region occupancy is reduced as the performance bottlenecks are removed, so it is a good idea to do scheduling by class. Fewer transaction classes make it easier to balance the workload by changing the number of message regions when necessary.

The following items contribute to higher than normal region occupancy:

- A system that is processor constrained, or one where IMS has low priority, causes MVS to allocate insufficient CPU resource to IMS and its dependent regions. This means that service times are longer, and this in turn tends to increase the region occupancy and cause fewer work to be scheduled.

- Contention for IMS resources among dependent regions can increase occupancy for all regions (see 5.2.5, “Number of Message Processing Regions” on page 113).

- A system that has a high paging rate also tends to have higher region occupancy. As the region receives less service while waiting for page-fault resolution, additional pages allocated to that region may be stolen, which in turn lengthens the time-in-system for the transaction being processed.

- Program loading of application program subroutines or subprograms can cause contention on the program libraries, and again this leads to greater region occupancy.

5.2.4 Mean I/O IWAIT Times

I/O IWAIT time is typically the largest single contributor to transaction time-in-system. There are two obvious ways to decrease I/O IWAIT time:

- Reduce the number of I/Os
  - See Chapter 8, Understanding IMS Use of Real Storage and Pools.
  - See Chapter 15, Performance of IMS with DB2.

- Reduce the duration of I/Os
  - See Chapter 7, Understanding Problems with the I/O Subsystem.

Longer than expected IWAIT times can be caused by channel, control unit, or device contention. The relative dispatching priority of the region issuing the I/O is another contributing factor. IWAIT time includes the time from issuing the I/O request until the dependent region is dispatched after the I/O is complete.

IWAIT time greater than 300 ms is usually caused by interference from shared DASD or severe overcommitment of a control unit, head of string, internal path, or device.

If the dependent region is not redispached after an event is started because of CPU contention, long IWAIT times can occur. The IWAIT event is posted complete only when the dependent region resumes execution.
### 5.2.5 Number of Message Processing Regions

The number of message-processing regions has a direct effect on transaction response time. If too few regions are available, the transactions may remain on the input queue for a long period of time. Conversely, if too many regions are processing the same application workload, there may be contention for the database records (PI enqueues). Therefore, we recommend that the number of message-processing regions be carefully monitored to ensure that there are no queue build-ups.

In environments with fewer transaction types, higher levels of occupancy can be tolerated because the application programs may be able to remain in the regions and process multiple transactions without causing other transactions to queue.

Specify a minimum number of regions that is consistent with achieving the response time objectives. Increase the number of regions only when the workload rises above what the current number of regions can process and still achieve the expected response time.

Having too many message-processing regions can increase CPU consumption by reducing the number of transactions per scheduling. This in turn causes more schedules and program loads. Too many regions also demand more storage, both for the regions themselves and for pools such as the IMS PSB pool and ENQ/DEQ pool. They also potentially increase contention among regions for IMS buffer pools, IMS data sets, enqueued resources (PI), internal IMS latches, and program load libraries.

The purpose of this section is to ensure that the system is not executing with too few or too many message regions. Reducing the number of dependent regions reduces resource contention and, conversely, reducing the resource contention may reduce the required number of message regions.

The number of message processing regions required by IMS can be calculated by multiplying the transaction arrival rate by the transaction time-in-system, and dividing by the desired region occupancy:

\[
\text{# Regions} = \frac{\text{Arrival Rate} \times \text{Time – in – System}}{\text{Region Occupancy}}
\]

**Example:** If the arrival rate is three transactions per second and the time-in-system is one second, how many regions are necessary for 50% region occupancy? Calculate the number as follows:

\[
\text{# Regions} = \frac{3 \times 1.0}{0.50} = 6
\]

If the time-in-system is reduced to 0.5 second, the number of regions is reduced to 3:

\[
\text{# Regions} = \frac{3 \times 0.5}{0.50} = 3
\]
Regions dedicated to WFI transaction processing are not included in this calculation because their region occupancy is 100%. The same holds true for BMPs. However, in the newer IMSPARS, the resource availability report contains a field called SUBQ6TIME. This field records the time that a scheduled program in a region is waiting for input. The actual occupancy, thus, can be figured out by region.

These examples illustrate the importance of reducing transaction time-in-system. It is difficult to obtain the actual transaction arrival rate, and time-in-system includes added time caused by resource contention; therefore, estimates must be used to arrive at the required number of regions.

Figure 40 on page 115 gives the formula with time-in-system expanded into specific components. To estimate the number of message regions necessary for an IMS system, the expected transaction rate can be substituted for T, and the guideline numbers can be used for schedule time (4 ms) and program load time (200 ms, or 20 ms if using VLF/LLA). To estimate program elapsed time, multiply the number of I/O IWAITs per transaction by 21 ms (for the 3390). Use actual measured values, if available, rather than guideline numbers.

Do not include regions dedicated to WFI transactions or BMP regions in this calculation because, again, their region occupancy is 100%. Subtract the number of these types of transactions and IWAITS from the totals on the program summary report.

These values are gathered from the run profile, program summary, and region summary reports.

The number of message regions determined by this method can be attained if critical resource bottlenecks are removed as prescribed in the sections that follow, and if transaction class scheduling is not severely restricted.
\begin{equation}
\text{\# Regions} = \frac{(1 + N) \times (T \times (S + PL + (I \times DT) + (1.1 \times C)))}{\text{Percent Region Occupancy}},
\end{equation}

where

\(N = \text{System Interference (default = 15 percent of elapsed time)}\).

\(T = \frac{\text{Total Number of Transactions Processed}}{\text{Monitor Trace Interval in Seconds}}\)

\(S = \frac{\text{Program Summary Schedule Elapsed Time}}{\text{Number of Schedules}}\)

\(PL = \frac{\text{Region Summary Program Total Mean Schedule End to First DL/I Call}}{\text{Total Number of Transactions Processed per Total Number of Schedules}}\)

\(I = \frac{\text{Total I/O IWAITs (Program Summary Report)}}{\text{Total Number of Transactions Processed}}\)

\(DT = 21 \text{ for device type 3390}\)

\(C = \text{CPU Time per Transaction (Program Summary Report)}\).

\(\% \text{ Region Occupancy: Use desired percentage of region occupancy (normally 60-65%).}
\quad \text{Use actual when solving for N.}\)

Figure 40. Formula for Determining the Number of Message Regions

### 5.2.6 Application Program Mean Elapsed Time

Performance problems caused by application design normally manifest themselves as a high number of DL/I calls per transaction, a high number of database I/Os per call, or long program elapsed times resulting from dynamic loading of subprograms or subroutines.

The application program elapsed times are calculated from the first message DL/I call to the logging of the X’35’ log record (enqueue of the output message on the LTERM). The processing within the region can be divided into the time spent in DL/I processing and the time spent outside DL/I processing.

The time outside DL/I processing can be spent in application code execution, subprograms loading, page faulting, interruption of CPU service by higher priority tasks, or I/O time to OS data sets owned by the application programs.

Within DL/I processing, the IWAIT time is recorded separately. This is discussed in more detail in 4.2.6, “DL/I Call Summary Report” on page 102.
If program elapsed time is higher than expected, and the database I/Os are within the guidelines, then the problem is somewhere outside of IMS control.

Possible areas to analyze are:
- Logging contention
- CPU overcommitment
- Paging
- PI contention
- I/O contention
- Application program usage

Mean program elapsed times of 1 to 2 seconds or more may be acceptable for some applications, but too high for others. The 1 second guideline is derived from approximately 30 to 40 database calls. If mean program elapsed times are less than one second and there are not any high maximums (longer than 3 seconds), the application program design is probably not the cause of major IMS performance problems. However, it is recommended that you refer to Chapter 12, “IMS Application and Database Design for Performance” on page 279 to review your applications in an attempt to lower the average transaction profile.

5.2.7 Implementation of Performance Recommendations

Analysis of an IMS system using the charts from Chapter 11 will probably result in a list of recommended changes to the system. Some recommendations will be easier to implement than others, and some will have a greater impact than others. The more significant bottlenecks must be eliminated to obtain noticeable performance improvement.

It is not practical to implement one change at a time and to measure the effect of each change. Many times, recommended changes depend on one another for their overall effect.

Once a set of recommendations has been implemented, run the IMS monitor and check the key performance indicators to determine whether the critical resources need to be analyzed again. If performance problems still exist, or the system needs finer tuning, start again at the beginning.
Chapter 6. Understanding CPU Resource Problems

If IMS is getting the CPU service when work has to be done, CPU is not a performance bottleneck. This chapter addresses how to improve IMS performance by increasing the CPU resource available to the IMS control region.

6.1 CPU Problem Identification Chart

Figure 41 Identifies CPU resources related to IMS performance.

Figure 41. CPU Performance Identification

6.2 Dispatching Priority:

Dispatching priority has a number of aspects that can be manipulated.
6.2.1 IMS Control Address Space

The job dispatching priority dictates in what sequence address spaces are to be dispatched when they have work to do. Any job with a dispatching priority higher than IMS and its dependent regions can cause interference. MVS tasks, such as the Master Scheduler, JES, VTAM, and GRS, normally run at higher dispatching priorities. RMF and GTF must run at a higher dispatching priority than IMS to obtain the service necessary to collect the correct data.

If the system paging rate is higher than 30 page-ins per second, IMS could be the highest priority task after the MVS system tasks and still not receive enough service. In this case, main storage utilization should be evaluated (see 8.3, "System Paging" on page 139).

Figure 42 on page 119 has a suggested dispatching priority assignment.

6.2.2 IMS Control Region

If the IMS schedule NOT-IWAIT time (Figure 26 on page 83) is greater than 4 ms and the total system paging rate is low (fewer than 12 pages per second and a page resolution time of less than 100 ms), higher priority tasks may be executing in the system and using the CPU cycles needed by IMS. Determine whether these higher priority tasks must run at their present dispatching priority. Lowering their DPRTY below that of IMS, if possible, gives the CPU resource back to IMS.

6.2.3 IMS DLS Address Space

Fix the dispatching priority of the DLS address space at one level below that of the IMS control region. The reason is that the DLS address space processes the open/close and allocation of all full-function databases. Fast Path databases are owned by the control region. Additionally, the scheduling of transactions is performed in the DLS address space if it loads the application PSB.

6.2.4 DBRC Address Space

Fix the dispatching priority of the DBRC address space at least one level above those of the IMS DLS and control region. IMS uses DBRC to process the database opens, first calls to the database, EOV on a database, the database close, and all log data set changes. The higher dispatching priority ensures that DBRC processing occurs when necessary.

6.2.5 IRLM Address Space

The IMS Resource Lock Manager (IRLM) address space is required for N-way data sharing, and is optional for single-system program isolation usage. The dispatching priority of the IRLM address space is recommended to be the highest possible. This is, normally, DPRTY=(15,15) in the startup procedure.

Most use of the IRLM takes place under the dependent region TCBs, using cross-memory services. However, the occasional task that executes within the IRLM address space itself must have top priority within the IMS system (for example, deadlock detection). IRLM must also be higher than any other MVS clients such as VTAM or VSAM might be using. If this higher dispatching priority is not specified, response times are lengthened.
6.3 IMS Dependent Regions

Specifying a unique dispatching priority ensures that a dependent region is always in the same position on the MVS dispatcher sequence queue relative to the IMS control regions and dependent regions.

Without unique dispatching priorities, all dependent regions may have the same dispatching priority assigned to them. Therefore, the message regions may be dispatched in different priority sequences each time they are started. If the dependent regions are placed in either the mean-time-to-wait group or the rotate priority group, they can be at different relative dispatching priorities at any given time during their execution. The rotate type of dispatching is usually adequate for dependent regions that process the same scheduling classes. If the scheduling of dependent regions is by class according to type of work, a unique dispatching priority is a better alternative.

Not only is it important that the IMS control region get CPU service when needed, but as a result of parallel DL/I, the same is true for the dependent regions. The DLS address space requires a dispatching priority just below that of the control region. If other tasks on the MVS dispatcher sequence queue come between the IMS control address spaces and the dependent regions, the intervening tasks receive service before the dependent regions. Therefore, raising the DPRTY of the dependent regions higher than that of the intervening jobs may improve transaction response time.

The message processing regions need a higher dispatching priority than the BMPs, but the BMPs must immediately follow them as the BMPs contend for the same IMS software resources (buffer handler latches, program isolation enqueues) as the message regions. A BMP could be interrupted while holding an IMS software resource and, therefore, suspend either the control address spaces or the MPP’s execution. The address spaces needing the resource wait until the BMP is redispatched and the resource is released. Tasks intervening between MPPs and BMPs could cause dependent-region elapsed time to increase, even though the dependent regions are getting adequate CPU service.
6.4 IMS Latch Statistics

The basic serialization of IMS tasks is controlled by ownership of the IMS latches. When IMS tasks are processing, they compete for ownership of these latches. If a latch is owned by one task and another task requests it, the requestor is placed in a wait state. When the holder of the latch frees it, all the tasks waiting for that latch are posted, and the next requestor then acquires the latch. (This is normally the requestor with the highest dispatching priority, but it could be the user who just released the latch if the program requests the latch again before an IWAIT is issued.) Figure 43 on page 122 is a sample of the IMS monitor latch conflict statistics report.

The latches and their functions are as follows:

<table>
<thead>
<tr>
<th>Latch</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP</td>
<td>System dispatcher, not used</td>
</tr>
<tr>
<td>DCSL</td>
<td>DC Checkpoint DC system latch, formerly part of DFSLATE0</td>
</tr>
<tr>
<td>LUML</td>
<td>DC LU 6.2 LUM serialization</td>
</tr>
<tr>
<td>CONV</td>
<td>DC conversation checkpoint, formerly part of DFSLATE0</td>
</tr>
<tr>
<td>TERM</td>
<td>DC terminal, added with ETO</td>
</tr>
<tr>
<td>LUBT</td>
<td>DC LU 6.2 LUB-TIB control block chain</td>
</tr>
<tr>
<td>LRLT</td>
<td>Log router latch</td>
</tr>
<tr>
<td>SCHD</td>
<td>TM scheduling (old block mover)</td>
</tr>
<tr>
<td>TCTB</td>
<td>TM TCT individual table latch</td>
</tr>
<tr>
<td>APSB</td>
<td>TM allocate PSB, block mover</td>
</tr>
<tr>
<td>PDRB</td>
<td>TM PDIR block, block mover</td>
</tr>
<tr>
<td>PSBP</td>
<td>TM PSB pool, block mover</td>
</tr>
<tr>
<td>DMBP</td>
<td>TM DMB pool, block mover</td>
</tr>
<tr>
<td>PSBB</td>
<td>TM PSB block, block mover</td>
</tr>
<tr>
<td>DMBB</td>
<td>TM DMB block, block mover</td>
</tr>
<tr>
<td>PDRP</td>
<td>TM PDIR pool, block mover</td>
</tr>
<tr>
<td>DBAU</td>
<td>TM DBRC authorization, block mover</td>
</tr>
<tr>
<td>DDRB</td>
<td>TM DDIR block, block mover</td>
</tr>
<tr>
<td>DDRP</td>
<td>TM DDIR pool, block mover</td>
</tr>
<tr>
<td>DBBP</td>
<td>Used to serialize OSAM database buffer pool requests; formerly DFSFUNL0</td>
</tr>
<tr>
<td>DBLR</td>
<td>DB DFSDBLR0 module</td>
</tr>
<tr>
<td>SUBQ</td>
<td>TM subqueues</td>
</tr>
<tr>
<td>DBSL</td>
<td>DB checkpoint; formerly DFSLATE0</td>
</tr>
<tr>
<td>USER</td>
<td>DC user ETO</td>
</tr>
<tr>
<td>DBLT</td>
<td>RSR sharing latch</td>
</tr>
<tr>
<td>CCTL</td>
<td>System DBCTL resource; formerly DFSFUNL0</td>
</tr>
<tr>
<td>VTCB</td>
<td>System CBTS VTCB pool ETO</td>
</tr>
</tbody>
</table>
VLQB    System CBTS LQB pool ETO  
CBTS    All CBTS for control blocks that are used dynamically; formerly DFSFUNL0  
BLKM    TM SMB LU 6.2 queue hash table  
QMGR    System queue manager  
QBSL    System queue buffer  
SMGT    System storage management  
DBLK    Used to free dependent-region control block storage  
XCNQ    DB, exclusive control of enqueue or dequeue; used for program isolation  
ACTL    Statistics logging; used to serialize the IMS monitor logging  
LOGL    Logical logger; used to serialize IMS logging  

If the LOGL latch seems to have a high number of contentions, analyze the allocation and I/O rate to the OLDS. Refer to the IMSPARS logical logger statistics (Figure 114 on page 355).

Take into account that, as the number of dependent regions increases, the contention for the LOGL latch also increases. Normally, this latch is held only while the data is moved into the log buffer and does not affect response time. Response time can be affected by longer I/O time on the WADS and possibly the OLDS. This latch is of concern only after all the other bottlenecks in the IMS MVS system have been eliminated.

Normally, the ACTL latch contention is highly unlikely. However, if the contention begins to increase over a period of time, a possible solution is to turn off the monitor for a period of time.

Very little can be done with the other latches in the report.
### LATCH CONFLICT STATISTICS

<table>
<thead>
<tr>
<th>LATCH NAMES</th>
<th>CONTENTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP</td>
<td>0</td>
</tr>
<tr>
<td>DCSL</td>
<td>0</td>
</tr>
<tr>
<td>LULM</td>
<td>0</td>
</tr>
<tr>
<td>CONV</td>
<td>0</td>
</tr>
<tr>
<td>TERM</td>
<td>0</td>
</tr>
<tr>
<td>LUJT</td>
<td>0</td>
</tr>
<tr>
<td>LRLT</td>
<td>0</td>
</tr>
<tr>
<td>SCDT</td>
<td>0</td>
</tr>
<tr>
<td>TCTB</td>
<td>216</td>
</tr>
<tr>
<td>APSB</td>
<td>0</td>
</tr>
<tr>
<td>PDRB</td>
<td>6</td>
</tr>
<tr>
<td>PSBP</td>
<td>0</td>
</tr>
<tr>
<td>DMDB</td>
<td>0</td>
</tr>
<tr>
<td>PSBB</td>
<td>0</td>
</tr>
<tr>
<td>DMHB</td>
<td>0</td>
</tr>
<tr>
<td>PDRP</td>
<td>1</td>
</tr>
<tr>
<td>DBAU</td>
<td>0</td>
</tr>
<tr>
<td>DDRB</td>
<td>0</td>
</tr>
<tr>
<td>DDRP</td>
<td>0</td>
</tr>
<tr>
<td>DBBP</td>
<td>0</td>
</tr>
<tr>
<td>DBLR</td>
<td>0</td>
</tr>
<tr>
<td>SUBQ</td>
<td>42</td>
</tr>
<tr>
<td>DBSL</td>
<td>0</td>
</tr>
<tr>
<td>USER</td>
<td>0</td>
</tr>
<tr>
<td>DBLT</td>
<td>0</td>
</tr>
<tr>
<td>CCTL</td>
<td>0</td>
</tr>
<tr>
<td>VTCB</td>
<td>17</td>
</tr>
<tr>
<td>VLQO</td>
<td>1</td>
</tr>
<tr>
<td>CBTS</td>
<td>12</td>
</tr>
<tr>
<td>BLKM</td>
<td>0</td>
</tr>
<tr>
<td>QMGR</td>
<td>24</td>
</tr>
<tr>
<td>QBSL</td>
<td>32</td>
</tr>
<tr>
<td>SMGT</td>
<td>18</td>
</tr>
<tr>
<td>DBLK</td>
<td>0</td>
</tr>
<tr>
<td>XCNQ</td>
<td>102</td>
</tr>
<tr>
<td>ACTL</td>
<td>0</td>
</tr>
<tr>
<td>LOGL</td>
<td>3055</td>
</tr>
</tbody>
</table>

**Figure 43. IMS Monitor Latch Conflict Report**
Chapter 7. Understanding Problems with the I/O Subsystem

I/O resource contention results in longer I/O wait times per I/O request. Problems causing long wait (IWAIT) times are discussed in this section. The number of I/O requests issued depends on the size of various IMS buffer pools (see 9.1, “IMS Buffer Pool Storage” on page 161). The logic chart below identifies the five major I/O resource problem indicators. If any of the questions are answered yes, go to the detailed chart for further explanations and recommendations.

7.1 I/O Problem Identification Chart

The following chart, Figure 44, shows four ways to identify potential and actual I/O problems.

```
1. Does any DASD channel path have a utilization of 50% or more? Yes
   CHANNELS 7.2

2. Is any DASD more than 50% busy, or have an IOSQ time greater than 4 ms or a high activity count? Yes
   DEVICES 7.3

3. Are any maximum I/O IWAIT times greater than 300 ms? Yes
   I/O MISC. 7.4

4. Is any schedule end to first DL/I call mean time greater than 200 to 300 ms? Yes
   IMS MESSAGE REGIONS 7.6

5. Return.
```

Figure 44. I/O Performance Indicators
7.2 Channels

As the percentage of channel busy time increases, the probability also increases that an I/O request to a device on that channel will have to be queued. Each time an SSCH request cannot be issued and the request must be queued, the I/O wait time of the requesting program increases. Experience has shown that a channel that is greater than 30% busy can affect overall system performance and increase IMS response times.

7.2.1 Channel Busy Chart

The following chart, Figure 45, gives you three ways to diagnose channel-busy problems.

![Channel Busy Chart Diagram]

Figure 45. Identifying Channel-Busy Problems

7.2.2 Channel Imbalance

If only some of the DASD channels are greater than 50% busy, a channel imbalance exists. The channel imbalance can be corrected by rearranging the DASD volumes across the channels. For example, if all the IMS volumes are on Channel 1, which is 47% busy, and the MVS system and paging volumes are on Channel 3, which is only 10% busy, exchanging some of the high-activity IMS volumes on Channel 1 with some of the low activity volumes on Channel 3 balances the channel load and reduces the I/O wait times for the IMS databases.
Note

A channel with data sets used only by batch jobs may be 90% utilized and still provide adequate batch performance. On nondedicated IMS systems, it is important to isolate, if possible, the IMS online data sets from the concurrently used batch data sets.

7.2.3 Channel I/O Activity

If any of the DASD channels are more than 30% busy, a way to improve I/O wait time within the present hardware configuration is to reduce the actual number of SSCH requests issued. One item to consider is the block sizes of non-IMS sequential data sets. Data sets with large block sizes use fewer physical I/Os to process the same amount of data than data sets with smaller block sizes. It takes less time to transfer an additional 4 KB of data than to reissue the SSCH for the additional data.

The minimum block size for IMS DASD data sets should be larger than 4 KB. Increasing the data set block size reduces the number of physical I/Os and CPU cycles needed to perform the I/O. It does not significantly increase the system paging rate unless the size of the data set buffer pool increases the job’s working set to a point where real storage becomes constrained. Review the main storage resource (see section 9.1, “IMS Buffer Pool Storage” on page 161) as well when attempting to reduce the number of system I/Os.

7.3 Device Activity

Even though the channels on a system may not be overloaded, a DASD that is more than 30% busy, or has an IOSQ time greater than 4 ms, or has a high activity count can cause long I/O wait times. High utilization of a DASD can occur if there is one or more high-usage data sets, or wide seek patterns across the volume. Refer to the RMF Monitor I direct access activity report (Figure 12 on page 48), column “Avg Number Alloc,” to determine how many data sets are opened on the DASD volume. The chart shows in Figure 46 on page 126 gives you five ways to diagnose channel imbalance problems.
7.3.1 Device I/O Activity

The level of activity on a device is a result of the total number of physical I/O requests issued to data sets on the device and the time each I/O takes to complete. The total number of physical I/Os can be high as a result of one or more high-usage data sets on one volume. Where multiple high-usage data sets reside on the same volume, moving some of these data sets to volumes that are not as busy can help to decrease the level of activity on the busy volume.

For a single high-usage data set on a device, the solution is not so simple. If the data set is an IMS database, reorganization could reduce the number of I/Os. If only one non-IMS job is referencing the data set, high utilization is not necessarily a problem as long as the volume is isolated from other I/O contention problems, such as a busy channel or control unit. The data set could be reallocated and split across multiple volumes. To force the split, all of the space on the volumes referenced for the data set allocation must have been previously allocated, with the exception of the amount of space to be used by the data set.
7.3.2 Data Set Placement and Allocation

Data set placement on a DASD with multiple data sets greatly affects the duration of the I/O. Wide seek patterns increase the duration of I/Os. It is recommended that volumes with multiple non-VSAM data sets that are referenced by many jobs (many opens/closes issued) have the volume table of contents in the center and the most frequently referenced data sets surrounding it. Free space between data sets is undesirable. Periodically reallocate and reorganize data sets with secondary extents to eliminate the secondary allocations. Place the data sets with the highest reference activity contiguously on the pack.

Allocate all data sets by cylinders. If a data set is allocated by tracks, every I/O request to that data set requires track verification. Track verification ensures that the next track referenced is contiguous to the previous track; therefore, the cylinder address does not have to be updated. This type of allocation increases CPU overhead for each I/O request. Cylinder-allocated data sets perform cylinder verification only after the I/O requests data from the last track on the cylinder.

7.3.3 Shared DASD

Shared DASD can cause device contention. If a device is busy processing an I/O request from another system, the first SSCH request issued to the device by this system results in a device-busy return. All subsequent requests are queued until the outstanding I/O from the other system is complete. If the device is on a shared control unit, a control-unit busy results if any other I/O request to a device on that control unit is in progress. For this reason, it is best to isolate IMS volumes on nonshared control units. If this is not possible, attempt to vary the path offline from the other system to the IMS volumes.

7.3.4 Control Unit, String, Device Busy

A device-busy condition can occur in a shared DASD environment if one system has an I/O data transfer request to any device on a control unit and a second system issues an SSCH request to the same control unit. It is best to use nonshared control units for high usage data sets whenever possible.

The RMF direct access device activity report can also be analyzed to determine whether contention exists on the path to the device. The average IOSQ time and the average PEND time fields can be analyzed to determine whether contention exists for the device.

7.4 Miscellaneous

Long I/O wait times result from either device or control-unit contention (see 7.3, “Device Activity” on page 125) or other interference in the system. This section addresses the events that increase the length of I/O wait times but which may not appear in the device activity indicators.

7.4.1 Shared DASD Chart

The chart in Figure 47 on page 128 addresses ways of fixing shared DASD problems.
1. If shared DASD is implemented, isolate IMS volumes on control units and vary these control units offline to the sharing system if possible.

2. Ensure that IMS data sets are on volumes mounted as PRIVATE to avoid contention with work data sets (TSO, SORT).

3. If VSAM KSDS CA or CI splits occur on an IMS data set, reallocate with more free space and reorganize the data set.

4. Return.

---

**Figure 47. Identifying Remedies to Shared DASD Problems**

---

### 7.4.2 Shared DASD: Control Units and Devices

Shared DASD can affect device activity, or I/O wait duration, or both. As previously shown, shared DASD can increase the number of waiting I/O requests on the device queue because the device, control unit, or head of string is busy processing a request from a sharing system. The length of time an I/O request is queued is part of the wait; therefore, the longer it takes to start an I/O successfully, the longer the wait time. Isolation of the IMS volumes on control units whose path to the sharing system is off line decreases I/O wait times.

### 7.4.3 Private DASD

DASD volumes that are not mounted as private to MVS can be used for temporary work data sets. These data sets can cause wide seek patterns and high I/O activity contention with the other permanent data sets on the volume.

Place IMS data sets on volumes that are mounted as PRIVATE to avoid contention with TSO and SORT work data sets.

### 7.4.4 VSAM Key Sequenced Data Set CI/CA Splits

The KSDS portions of IMS databases incur a large number of IWAITs when CI or CA splits occur. No more than a few control area splits are expected to occur in a day. To complete the processing of a CI/CA split, the number of VSAM I/Os increases and the number of instructions that VSAM requires to process these extra I/Os also increases. This increase of CPU cycles and I/Os contributes to a longer elapsed time for the transaction.

With cylinder CAs, CA split is essentially the writing and rewriting of a cylinder of data. The VSAM catalog can be monitored with the AMS utility LISTCAT to
determine whether CI/CA splits are occurring. These fields in the VSAM catalog are updated when the data set is closed; therefore, the LISTCAT needs to be done after the data set is closed. Once the occurrence of CI/CA splits is noted, reallocate the database with more free space and reorganize the KSDS data set with IMS utilities to stop the splits and reduce the number of IWAITs. If the above recommendations do not eliminate the number of CI/CA splits, a database or application design review may be necessary to determine whether a different organization of the database or a different application processing sequence can be implemented to reduce the number of CI/CA splits. This may be difficult with an index whose records are always added at the end; for example, date, account number, and order number. Reorganize this type of database regularly, and do not specify free space when loading the database.

7.4.5 Monitoring Procedures

If all of the above areas have been investigated, and some I/O maximum IWAIT times are still greater than 300 ms, the problem could be that the CPU is overcommitted and the dependent regions cannot be dispatched in order to post the IWAIT complete. Another item that could lead to erratic IWAIT time is paging in the system. If a page fault is taken after the IWAIT is issued but before the I/O is started, or if a page fault is encountered after the I/O is complete but before the IWAIT is posted as complete, elapsed time for the event could be longer.
7.5 IMS Data Sets Chart

Figure 48 shows IMS system data sets performance problems.

1. Check the mean IWAIT times.
   a) OSAM - if mean IWAIT time is greater than guidelines, make sure write check is turned off (Remove DCB=OPTCD=W) or check path interference.
   b) VSAM KSDS - if mean IWAIT time is greater than guidelines,
      1) Imbed sequence sets with the data.
      2) Turn off write check if it is on, with AMS DEFINE statement.
   c) VSAM ESDS - if mean IWAIT time is greater than guidelines, make sure write check is turned off (DEFINE).
   d) System Data Sets - Check I/Os per second and mean I/O times. Increase pool size or check data set placement.

2. If mean I/O IWAIT times vary for the same database in different dependent regions, check the dependent region dispatching priority.

3. If none of the above seems to be true, check again.

4. Return.

Figure 48. Addressing IMS Data Set Problems

7.5.1 Mean I/O IWAIT Times

Average mean I/O IWAIT times of more than 21 ms for a 3390 device indicate that some or all of the databases have a mean I/O IWAIT time that can be reduced. It is important to remember that IWAIT times include the actual I/O service time, plus the time in the MVS I/O supervisor waiting to issue the Start I/O, and the time from completion of the I/O event until IMS or the dependent region receives control from MVS. It is assumed that all hardware, shared DASD, allocation, and data set placement problems have already been solved. This section addresses what can be done to the access methods that process the IMS databases to reduce the IWAIT durations.
7.5.1.1 OSAM Data Sets
It is recommended that OSAM access to databases not exceed a 21 ms IWAIT time for 3390 device types. If these mean IWAIT times are greater than the guidelines, analyze the data set placement or dispatching priorities of the regions having problems.

If the above does not seem to be the problem, investigate the write check option. If the write check option is turned on, turn it off for that data set.

7.5.1.2 VSAM KSDS Data Sets
VSAM KSDS processes both the index portion and data portion. However, an IMS IWAIT is recorded for each I/O. Therefore, multiple IWAITs can be recorded for a single KSDS retrieval. It is best if IWAITs to a VSAM data set do not exceed 21 ms for 3390 device types.

A VSAM KSDS has multiple levels of indexes; the sequence set is the lowest level. Direct (as opposed to sequential) processing of a KSDS always causes a top-down search of the index component starting with the highest-level index record, through an intermediate-level index record, and finally to a sequence set record (assuming at least a three-level index). Given a three-level index structure, this means three separate I/Os, unless some portion of the index component is kept in main storage. Allocate a buffer for the highest-level index CI, the intermediate-level CIs, and one or more sequence set CIs. Additionally, it is recommended that you attempt to isolate all the index component CIs of all KSDSs from any data buffers. Buffers for the data and the index components can be allocated separately within the IMS VSAM buffer pools.

To find the number of buffers needed in a subpool to hold all of the high-level index CIs, use the VSAM catalog entry for the index component.

7.5.1.3 Other VSAM Parameters
The WRITECHECK parameter is an option for both VSAM KSDS and ESDS components. The validity check is the same as DCB=OPTCD=W for OSAM. Do not use WRITECHECK in IMS systems. WRITECHECK is specified in the DEFINE CLUSTER DATA or INDEX statements of VSAM access method services. The VSAM catalog entry for the data set contains the indicator that WRITECHECK is to be performed. Delete the WRITECHECK option using the AMS ALTER process for each data and index component in the cluster.

The RECOVERY option for the VSAM data set DEFINE statement was originally designed to allow a restart if an abnormal termination (abend) occurred while loading the data set. The option preformats a CA with free-space CIs and writes a software end-of-file in the first CI of the next CA. The data is written after the CA is preformatted. This results in two I/Os per CI during a load. No VSAM IMS utility is available that uses the RECOVERY option for restart of a VSAM load. RECOVERY is the default; override it by specifying SPEED.

7.5.1.4 Message Format Data Set
When I/O to the format data set is required, it usually involves a pair of I/Os (MID or MOD, then DOF or DIF). The design of the compiler tends to place block pairs adjacent to each other on the data set (same track or adjacent tracks). If a directory I/O is not required for the second block of a pair, arm movement is not required and the I/O is quite fast (about 15 ms).
Define the size of this pool large enough to contain the active formats in order to manage the I/O rate to the library. This I/O rate is expected to be less than 0.5 I/O per second or less than 0.5 I/O per transaction.

If a page fault occurs while processing a format in this pool, IMS issues a page load for that page and IWAITs the ITASK until the page is loaded into storage. This feature allows the IMS control task to continue processing and not suspend the processing of terminal requests and scheduling of transactions.

IMS builds a dynamic directory entry for each format the first time it is referenced, unless the format is in the $$IMSDIR member. This option eliminates the initial I/O for the directory entry on subsequent loads of the format if deleted from the pool. The user has the option to use the offline IMS utility DFSUTSA0. This option builds the resident directory entries ($$IMSDIR) at IMS initialization time and after each online change to the MFS library. Using the option allows a more consistent response time for the transactions as the format directory entries are resident in the pool on all references, including the first.

The specification of FRE is an important consideration for the efficient use of the MFS pool. Each FRE controls a format block that resides in the pool. Therefore, there must be an FRE for each format block that can be resident in the pool. To calculate the number of FREs required to manage the pool, use the following guideline:

- Divide the size of the MFS pool by the average size of the formats. To this result, add a factor of about 15% of the previous calculation.

After this calculation is done, monitor the pool to determine whether the free space from a /DIS POOL MFBP is consistently high. If enough FREs are defined, the free space in the pool should be relatively low, although it could vary if some large formats are loaded into the pool.

### 7.5.2 Dispatching Priority and I/O IWAIT Times

If mean I/O IWAIT times for the same database appear to be longer in one dependent region than in others, the dependent region may be getting noticeably fewer CPU cycles because it has a lower dispatching priority. The IWAIT times for the databases by region can be found on the IMS monitor region IWAIT report (Figure 25 on page 80). The IMS monitor IWAIT time is the duration of time between the IMS dependent region request for the I/O until the dependent region is redispached by MVS when the I/O is complete.

The time interval consists of not only the actual I/O time but also any time used by higher-priority tasks in the system. An I/O for a database in a dependent region with a dispatching priority of 245 could take 35 ms, while an I/O to the same database for a region with a dispatching priority of 180 could take 55 ms. The only difference between the two I/O requests is the length of time it takes MVS to redispach the second dependent region. The difference can vary, as it depends on the level of the CPU activity at the time the I/O was completed and the number of tasks that were active at a dispatching priority between the two message regions.

Because I/O times are the largest single factor in program elapsed time and subsequently response time, it is best if transactions that require consistently good
response time are processed by the dependent region with the highest dispatching priority.

If a program can execute in multiple regions, scheduling of the high-priority region cannot be guaranteed even if the class of the high-priority transaction is first in the JCL specification of classes. IMS arranges the PST queue by placing the PST that just completed at the bottom of the queue. IMS then tries to schedule any transaction in the first available region that has the class of the transaction being scheduled. This technique employs a round-robin type of region usage.

### 7.5.3 I/O Priorities

When using the system resources manager (SRM) to control workload balancing within the MVS system, it is also possible to allocate priorities to I/Os. When defining the SRM parameters, a choice must be made between the two possible priority schemes for I/Os. The choice is indicated in the IEAIPSxx member and is documented fully in the MVS/ESA Initialization and Tuning Reference Manual (SC28-1453). In summary, the two options are:

- **IOQ=PRTY**
  This option allows you to define the I/O priority for each address space. By default, the address space dispatching priority is also the I/O priority.

- **IOQ=FIFO**
  This option is the default and effectively specifies that there are no I/O priorities; I/Os are done in the order they are requested.

The default (FIFO) is generally the best choice for IMS systems.

Some customers use IOQ=PRTY, with I/O priorities aligned with the dispatching priorities. Since almost all database I/O activity is performed under the dependent region TCBs, this ensures that MPP I/Os get priority over BMP I/Os when contending for the same database.

There is, however, one undesirable consequence that applies to IMS systems that update DEDBs. DEDB updates are performed by OTHREADs, which run under SRBs in the IMS control region. Also, OTHREADs do chained write I/Os. When using I/O priorities aligned with dispatching priorities, the asynchronous DEDB chained writes execute with higher priority (control region priority) than the dependent-region read I/Os. This can cause erratic read I/O performance for MPPs and IFPs, especially if there are BMPs heavily updating the DEDBs.

If you are a Fast Path user who wishes to exploit I/O priorities, define the I/O priority sequence as:

- IFPs
- MPPs
- Control region
- DL/I separate address space
- BMPs

It is essential that the BMPs have a lower priority than the control region. Otherwise, the Fast Path buffer pool quickly fills up with updated buffers waiting on the lower priority OTHREADs. This is not a problem for BMPs. When the buffer pool is full, the BMPs are unable to proceed, and the OTHREAD I/Os then execute.
In other words, the BMP processing is governed by the buffer pool rather than by the DASD itself. However, the IFPs and MPPs are also unable to get buffers and have to wait on the BMP writes (possibly to unrelated DEDBs).

This could also be a problem if the MPPs and IFPs are themselves heavy updaters. In this case, you want the writes to happen with at least as much priority as the reads, so that space in the buffer pool is created as fast as it is taken up.

Clearly, this aspect of Fast Path performance is difficult to predict and control. So for Fast Path users, the basic recommendation is to use IOQ=FIFO.

Note

Users of the MVS Workload Manager operating in Goal mode have no choice about I/O priorities.

7.5.4 IMS Data Set Placement

Some tuning considerations for data set placement need to be evaluated when monitoring the performance of an IMS system. The placement of the IMS system data sets at installation time is extremely important in avoiding data set contention. The guidelines for IMS system data sets are:

- Separate MVS paging and swapping volumes from the IMS data set volumes.
- Place PGMLIB, ACBLIB, OLDS, and the WADS on different control units and channels if possible.
- Separate the WADS from PGMLIB and ACBLIB.
- Do not place IMS volumes on control units shared with TSO.
- Mount all IMS volumes as private.
- Separate LGMSG and SHMSG data sets. A maximum of ten different data sets can be allocated for each; therefore, a high level of separation can be achieved. QBLKS is not that important as it is not very active.

7.6 IMS Message Regions

I/O resource contention increases IMS message-region program elapsed times, and if standard program fetch is used, program load times can also be lengthened.

When using standard program fetch, the PGMLIB is one of the most critical IMS data sets because it is referenced every time an application program or subroutine is loaded. Place the PGMLIB on a device where contention is minimal, and make the block size as large as possible to minimize the number of I/Os required to load a program. If the PGMLIB is a very active data set, place it on a cached DASD volume. The library can also be split into multiple copies of itself, one for each dependent region or set of dependent regions. Allocate these cloned libraries on different internal paths and channels if possible.

We do, however, recommend faster techniques for loading programs involving fewer or no I/Os, such as the use of VLF/LLA or virtual fetch or preload. These are discussed in detail in 10.8, “Program Load Options” on page 221.
Other factors that account for long schedule-end-to-first-DL/I-call times are message region dispatching priorities (see 6.3, “IMS Dependent Regions” on page 119), application program initialization, and the number of I/Os issued between schedule approval and the first DL/I call (see 10.7, “Schedule End to First DL/I Call” on page 221). These I/Os can result when the application program issues an OPEN for a user-defined MVS file.
Chapter 8. Understanding IMS Use of Real Storage and Pools

The amount of I/O activity on the system is determined by the availability of real storage. An overcommitment of real storage increases the paging rate. In the case of IMS, or any online response-oriented system, the paging rate and where the page faults occur have an effect on response times. The number of buffers and block sizes for data transfer have a direct relationship to the amount of I/O activity.

Optimizing the size of IMS pools becomes a trade-off between allocation of virtual storage to hold the data, and I/Os to retrieve the data. Virtual storage is the better choice for commonly retrieved data when real storage is available. When real storage is overcommitted, the system is forced into paging.

8.1 Storage/Pools Problem Identification Chart

Figure 49 on page 138 shows four ways of dealing with storage or problems.

Over the past few releases, IMS has implemented the use of the MVS PGLOAD facility in several areas. For example, when an input message is received in a RECANY buffer, the corresponding IMS subtask (ITASK) issues a PGLOAD for the appropriate terminal control block (VTCB). If the required page is not in real storage, the ITASK IWAITs, but the IMS TCB (for the control task in this case) continues processing another ITASK while the page is read in from DASD.

In cases where PGLOAD is not exploited, page faulting in the IMS control region or DL/I separate address space suspends processing of the relevant control task together with the dependent region being serviced by that control task. It can also delay processing in other dependent regions; for example, if the suspended task is holding a latch.

Therefore, as a general rule, it is better to perform data I/O than to page fault in the IMS control region address space, the IMS common storage area, or the DL/I separate address space (DL/I code, control blocks, and pools). We recommend that IMS pools be small and page fixed rather than large and pageable if the larger pools result in page faulting. If page faulting must occur in the IMS subsystem, try to limit it to the application programs where possible.

The arrival rate of transactions determines the page-fixing requirements for the IMS subsystem. For systems with a very low transaction arrival rate and a demand page-fault rate greater than 12 pages per second, it may be necessary to implement MVS storage isolation for the control region and the DL/I separate address space to achieve acceptable response time. Consider the database recovery control (DBRC) and the IMS resource lock manager (IRLM) address spaces, if present, for protection by the MVS storage isolation feature. Transaction arrival rates fluctuate in most systems. Therefore, it may be advisable to page fix certain virtual storage areas in CSAs, in addition to using storage isolation, when response time is a major objective.

Specific tuning variables and guidelines are provided in Chapter 3, “Monitoring the IMS Environment” on page 35.
Implement storage isolation instead of page fixing in the control region private area and the DL/I separate address space (DLS) private area. Storage isolation gives you the ability to protect the working sets of different applications or the CSA and is better than page-fixing IMS modules and control blocks to achieve acceptable response times.

Refer to 9.1, “IMS Buffer Pool Storage” on page 161 for detailed information on IMS page fixing of pools and control blocks and 8.9, “Storage Isolation” on page 158 for detailed information on MVS storage isolation.

8.2 Page Faulting

Page faulting in an online system directly increases the transaction time-in-system by the duration of the paging activity. The average I/O service time for a page in is 30 to 35 ms in a dedicated IMS system without I/O resource contention.
A page fault in the IMS control task is more severe because it suspends processing in the control task and in any dependent region waiting to be serviced by the control task. In conjunction with this, a page fault in the DLS address space can cause delays in processing DL/I calls, although the control task can still schedule and terminate regions while sending and receiving messages.

Depending on the page rate and paging I/O subsystem, the typical time required to resolve a page fault can range from 10 ms to 200 ms, with peaks over 500 ms. These delays in page resolution time can affect IMS performance dramatically. It is highly possible for IMS to wait 200 ms to be redispached after taking a page fault and then take another page fault because, while it was waiting, the SRM stole another page. As this is happening, the transaction response time can range from 2 seconds to 2 minutes or longer.

Therefore, it is essential that the working set of IMS be protected from page stealing as much as possible. This can be accomplished by implementing the storage isolation function of MVS/ESA (refer to 8.9, “Storage Isolation” on page 158).

8.3 System Paging

Figure 50 shows four ways of addressing system paging problems.

This section analyzes paging activity in an IMS/MVS system: how paging rates affect response times, and why it is important where page faults occur.

It is very difficult to give a rule of thumb about the levels of paging that can have an adverse effect on IMS response time. If sufficient expanded storage is available, paging should not cause a problem for the DLISAS or the dependent regions. If
DREF is used for pages moving only to expanded storage (in DFSDRFxx for VTCBs), a few pages per second may be acceptable for the control region; if DREF is not used, no paging is the goal. The rest of the discussion assumes that paging does incur actual I/O.

If the paging configuration consists of the 3990 Model 3 cache controller or the cached 9340, the paging rate can be 60 pages per second and still not affect IMS to a noticeable extent.

When the paging rate is high, it is necessary to attempt to minimize paging and control the location in storage where the page faults occur.

It is not so much the number of page ins per second that affect IMS response time but the amount of time it takes to resolve a page fault. This resolution time starts when IMS is suspended and ends when IMS is redispached to do useful work.

Take, for example, a system paging at 8 pages a second with a page resolution time of 250 ms versus a system paging at 12 pages a second with a page resolution time of 60 ms. The system with a page resolution time of 60 ms gives better response than the system with a time of 250 ms.

In the previous example, if IMS took two page faults a second with a page resolution time of 250 ms, it would be suspended for 500 ms, in contrast with 120 ms with a page resolution time of 60 ms. A high paging rate with a low page I/O service time could be better than a low paging rate with a high I/O service time. Nonetheless, a low paging rate with a low I/O service time is the target for an IMS system.

8.3.1 IMS Dependent-Region Swapping

IMS dependent regions are made nonswappable at region initialization time. Therefore, the IMS regions are always nonswappable unless your installation has modified the IMS code. The module DFSASK00 issues a TRANSWAP SYSEVENT SVC at region initialization to make the dependent regions nonswappable.

Note

The modification to make the dependent regions swappable might be done for low-usage test IMS systems that execute on a storage-constrained system. You should never make the dependent regions swappable in a production system.

8.3.2 IMS Batch DL/I Region Swapping

The user controls whether the IMS batch DL/I address space is swappable or nonswappable by means of the SWAP= parameter on the execute statement in the batch DL/I procedure.

Table 10 on page 141 indicates how this parameter is used in conjunction with the other parameters.
If the IEFRDER DD statement is coded as DD DUMMY, it is assumed to be present. If database block-level sharing is used by the batch program, IMS makes the batch program nonswappable.

<table>
<thead>
<tr>
<th>Table 10. IMS Batch DL/I Swapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SWAP=PARAM</strong></td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>N</td>
</tr>
</tbody>
</table>
| Omitted | Omitted | Nonswappable/swappable *

**Note:** * If the SWAP parameter is omitted and there is update intent, the address space is made nonswappable; otherwise it is swappable.

8.3.3 Paging Data Sets

Place the four paging data set types—pageable link pack area, common, local, and swap—on separate, nonshared volumes and control units. If more local or VIO paging data sets are allocated, place them on separate volumes as well. If the paging data sets cannot be on separate devices, reduce the number of data sets allocated. Two or more paging data sets on the same volume result in unnecessary device contention and arm movement. If separating paging volumes is a problem, it may be possible to reclaim one volume by allocating only one track to the PLPA data set and allocating the common storage area on the same device, with enough space to contain the overflow from the PLPA paging data set.

The optional duplex paging data set is used for recovery of I/O errors on the common paging data set. Duplex paging causes two I/Os for every CSA page out. If the option is not specified in the IEASYSxx member, IPLs are forced for the occasional I/O error, but only one I/O per CSA page-out is required.

Use the page data set for the swap option for systems with high TSO swap rates. If these swap data sets cannot be allocated on different strings and control units from the local paging volumes, it may be better to eliminate the swap data sets and let the swap pages page to the local page data sets.

In an IMS environment, additional real storage reduces real storage contention and out-performs high speed DASD. Placing paging data sets on fast devices with no contention reduces page resolution time, but the best answer to any IMS paging problem is additional real storage.
8.3.4 MVS Fix List

There is no requirement to fix MVS system modules in an MVS/ESA system. There may be a requirement to fix user modules. If you have such a requirement, specify the user modules in the IEAFIXxx member of SYS1.PARMLIB. For performance reasons, it is advisable to fix the resource cleanup modules. These modules are:

- DFSMRCL0 (IMS RTM Routine)
- ASUH2RMT (VSPC RTM Routine if present)
- ERBFRES (RMF RTM Routine)

These resource cleanup modules are in the fix list because they are loaded and referenced for every task termination in the system, and they can be found faster if they are in the MVS fix list. This results in saving CPU cycles.

8.3.5 IMS Batch DL/I BLDL Entries

If IMSESA.RESLIB is in the MVS LNKLST, and batch DL/I programs are executed or BMPs are run during the day, include IMS RESLIB in the list of libraries that the Library Lookaside (LLA) address-space controls. Refer to 10.8, “Program Load Options” on page 221 for additional information on this topic. Any modifications to IMS programs included in a library that is managed by the LLA address space can be refreshed without having to IPL the MVS system, as was necessary with the old MVS BLDL lists. The LLA address space can be refreshed without stopping the system; therefore, this option can reduce the elapsed time required to start and stop the BMP or batch DL/I programs.

8.4 IMS Paging

Figure 51 shows various IMS paging problems.

![Diagram showing IMS Paging Problems]

Figure 51. Identifying IMS Paging Problems
8.4.1 Storage Isolation of the IMS Control Region

All transactions that are processed require a certain percentage of the code in the IMS control region. This percentage is basically the working set of the IMS nucleus, device-dependent communication modules, MFS modules, and other IMS control modules. This working set size is the amount of storage to be specified for storage isolation to prevent page faulting for transaction processing. Individual system requirements vary depending on the IMS facilities used.

Implement the storage isolation function for the IMS control region. If this function is implemented, it is not necessary to page fix any percentage of the IMS nucleus. Storage isolation protects the referenced pages of the nucleus if the specification of number of frames is adequate.

Also, it is not necessary to page-fix the IMS DC control blocks, as they can be protected from page stealing by the storage isolation function. The storage isolation function allows the nonreferenced pages to be paged out; therefore, they do not hold any real storage frames as they would if page fixed.

For more detailed explanation of storage isolation see 8.9, “Storage Isolation” on page 158.

8.4.2 Storage Isolation of the DL/I Subordinate Address Space

Implement the storage isolation function for the DL/I separate address space using the LSO=S option. The storage isolation parameters specified must be different from the parameter specified for the control region. To achieve this, specify a unique performance group for the DL/I separate address space. This allows the referenced DL/I control blocks, pools, and code to remain in storage without being page fixed.

8.4.3 Storage Isolation of CSA

Normally, CSA in an MVS environment does not require storage isolation, but in some installations it may. To determine whether it is required, analyze the RMF Monitor I paging report (Figure 14 on page 52). If the CSA page-in rate is high (greater than eight pages per second with a page service time for the common page data set greater than 8 ms), consider storage isolation for CSA. The page service time for all of the page data sets can be found on the RMF Monitor I page data set report. This report is optional and is selected by specifying it in the RMF monitor startup procedure parameters.

8.4.4 IMS Page Fixing

An IMS system that is expected to maintain consistently good response time cannot afford to page fault in storage areas that affect all dependent regions. Page faulting in the IMS control region, DL/I address space (if LSO=S has been specified), IMS control blocks, and IMS buffer pools has a multiplier effect because it may delay all dependent regions. A minimum paging rate is allowable for things like command processing, checkpoint, or other seldom-referenced modules.

A high transaction arrival rate (more than 30 per second) is usually sufficient to keep the IMS control region working set in real storage. The page fixing options for IMS (DFSFIXxx) are necessary to provide satisfactory response times during periods of inactivity or low volume. Most small IMS systems fluctuate in transaction arrival rates, so it is advisable to page fix certain control blocks and pools and
invoke storage isolation for the control region and the DL/I separate address space. For low volume systems that have response-time objectives, it may also be necessary to fix commonly used DL/I modules by name.

Refer to 8.7, “Page Fixing IMS Pools and Control Blocks” on page 153 for a list of the pools that must always be page fixed; and 8.7.2, “Page Fixing IMS Control Blocks” on page 153, for control blocks that can be optionally page fixed.

8.5 IMS Control Blocks

This section describes the major IMS control blocks.

8.5.1 Overview

IMS maintains all of its information about the running system in IMS control blocks. IMS has several storage managers that it uses in an online environment to acquire storage from MVS. Each storage manager obtains areas of storage from MVS-called pools, and subdivides these as necessary to satisfy storage requests within IMS.

8.5.2 IMS Control Blocks Summary

The Table 11 table outlines the names of some of the more commonly used control blocks, and where the storage is allocated. For more information on MVS subpools, please refer to MVS/ESA SP V5 Diagnosis Guide and Reference (SC26-8848).

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Subpool</th>
<th>Above or Below the line</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAB</td>
<td>Archive anchor block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>ADSC</td>
<td>Fast Path area data set control block</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>AESL</td>
<td>Fast Path data entry database area data set list</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>AHDR</td>
<td>XTF autologon LU header/ autologon hash table synonym</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>AWE</td>
<td>Asynchronous work element</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>BCPT</td>
<td>Checkpoint ID table</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>BQEL</td>
<td>ISAM/OSAM/VSAM buffer queue element</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>BXQE</td>
<td>BCB queue elements for use by DFSBCBxx modules only</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>CBLK</td>
<td>Used in place of message queue to keep track of CPI-C schedule requests</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>CBT POOLS GLBL</td>
<td>CBT - Control block table pool Seems to be the storage for all these control blocks</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>CCB</td>
<td>Conversational control block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>CFEZ</td>
<td>Block used to store local parameters for DC trace reentrance</td>
<td>231</td>
<td>Above</td>
</tr>
</tbody>
</table>

Table 11 (Page 1 of 5). IMS Control Blocks
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Subpool</th>
<th>Above or Below the line</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLLE</td>
<td>Common latch list element</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>CM24</td>
<td>Below-the-line common work unit (CMWU)</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>CMWU</td>
<td>Common work unit</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>CSAB</td>
<td>Common services anchor block (see CSAG and CSAL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSAG</td>
<td>Common services anchor block (CSAB), global storage, in ECSA</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>CSAL</td>
<td>Common services anchor block (CSAB), local storage, in Extended Private</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>CULE</td>
<td>Common use list element</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>D1WA</td>
<td>Dispatcher work area</td>
<td>228</td>
<td>Below</td>
</tr>
<tr>
<td>DACT</td>
<td>DL/I call accounting block</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DBPB</td>
<td>Database purge block</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DBRC</td>
<td>DBRC work area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DCM</td>
<td>Deferred isolated log tracker control message</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>DDIR</td>
<td>Data management block directory entry</td>
<td>231/250</td>
<td>Above</td>
</tr>
<tr>
<td>DDRE</td>
<td>Data management block directory extension</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>DESC</td>
<td>LU 6.2 descriptor block</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>DG2W</td>
<td>Global dispatcher area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DL2W</td>
<td>Local dispatcher area</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>DMHR</td>
<td>VSO dynamic DHMRS for write staging area</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>DPST</td>
<td>Dependent-region partition specification table (PST)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DSME</td>
<td>Fast Path data space mapping entry. The data space mapping entry (DSME) maps the used portions of a Fast Path VSO data space. It is obtained during Fast Path open processing.</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>DSML</td>
<td>Fast Path data space mapping list. The data space mapping list (DSML) represents a Fast Path VSO data space. One is obtained whenever a data space is obtained at initialization and open.</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>EMAC</td>
<td>Extended DMAC. Contains globally accessible information regarding tracking for a particular area. Only remote IMS has an EMAC for each DMAC.</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>EPST</td>
<td>Fast Path extended partition specification table (PST)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>EQEL</td>
<td>In-doubt RIS EQEL</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>EZS</td>
<td>Initial block entry status</td>
<td>241</td>
<td>Below</td>
</tr>
<tr>
<td>FEIB</td>
<td>Front-end message switch interface block</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>FNGB</td>
<td>Fast Path notify control block</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>FPCP</td>
<td>Fast Path command parameter list</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>FSRB</td>
<td>Fast Path SRB</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Subpool</td>
<td>Above or Below the line</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>GDS</td>
<td>GAP descriptor</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>GESE</td>
<td>Global external subsystem entry</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>GIOB</td>
<td>Global I/O supervisor block (IOSB)</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>GOWA</td>
<td>Global OSAM work area (OSWA)</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>GQMW</td>
<td>Global queue manager work area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>GS24</td>
<td>24-bit quick block, global save area/asynchronous work element (AWE)</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>GSAV</td>
<td>global save area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>IAFP</td>
<td>IAFP data set control block</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>IDT</td>
<td>Identify table entry</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>IEQE</td>
<td>Inflight/indoubt data buffers</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>IOSB</td>
<td>I/O supervisor block</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>IRLM</td>
<td>IRLM parameter area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>KLSD</td>
<td>LSO dependent control block</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>L56X</td>
<td>Fast Path database control log record</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>LCLL</td>
<td>Local common latch list element (in extended private storage)</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LCRE</td>
<td>Local current recovery entry</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>LG24</td>
<td>Local save area (LSAV) below the 16 MB line</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>LGND</td>
<td>Logon descriptor control block sets</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>LGWA</td>
<td>Log work area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>LGWX</td>
<td>Log work area extension (private)</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LPST</td>
<td>Local partition specification table (PST) block</td>
<td>251</td>
<td>Below</td>
</tr>
<tr>
<td>LQB</td>
<td>Local queue block</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>LQMW</td>
<td>Local queue manager work area</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LRA</td>
<td>Log read area</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LS24</td>
<td>24-bit quick block, local save area/asynchronous work element (AWE)</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>LSAV</td>
<td>Local save area</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LUB</td>
<td>LUB pool below 16 MB line for LU 6.2</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>MPB</td>
<td>Milestone position block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>MSGP</td>
<td>Message pool (length of message and AWE &lt; 240)</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>OFB</td>
<td>Online forward recovery block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>OSWA</td>
<td>OSAM work area (IOMA)</td>
<td>228</td>
<td>Above</td>
</tr>
<tr>
<td>P128</td>
<td>General purpose 128-byte parameter list storage (31-bit private)</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>PCIB</td>
<td>Partition CIB</td>
<td>0</td>
<td>Above</td>
</tr>
</tbody>
</table>
### Table 11 (Page 4 of 5): IMS Control Blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Subpool</th>
<th>Above or Below the line</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDIR</td>
<td>PSB directory entry</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>PF62</td>
<td>Message prefix block for LU 6.2</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>PST</td>
<td>Partition specification table</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>QAB</td>
<td>QAB pool above 16 MB line for LU 6.2</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>QMBA</td>
<td>Queue manager buffer area</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>QSAV</td>
<td>Temporary save area set with parameter areas</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>RACW</td>
<td>RACF work area used for RACINIT, FRACHECK, and such.</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>RBAT</td>
<td>Fast Path VSO RBA update table entries. The fastpath VSO RBA update table entry maps a set of DEEB area RBAs that are written to DASD.</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>RCNT</td>
<td>Remote communication name table</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>RCTE</td>
<td>Fast Path routing code table entry</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>RECA</td>
<td>VTAM receive-any buffer</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>RPST</td>
<td>Restart partition specification table</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>RRE</td>
<td>Residual recovery entry</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>SAA</td>
<td>Stream archive anchor</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>SAP</td>
<td>Save area set prefix</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>SIDX</td>
<td>Subsystem index entry</td>
<td>241</td>
<td>Below</td>
</tr>
<tr>
<td>SLOG</td>
<td>DC monitor work area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>SMB</td>
<td>Scheduler message block</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>SOPB</td>
<td>Signon parameter block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>SRA</td>
<td>Stream receive anchor</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>SRBC</td>
<td>Common SRB pool</td>
<td>228</td>
<td>Below</td>
</tr>
<tr>
<td>STAT</td>
<td>DBCTL and DRA statistics area</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>STB</td>
<td>Stream block</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>STTR</td>
<td>DL/I trace stacks</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>SVPG</td>
<td>System service parameter lists (global - ECSA)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>SVPL</td>
<td>System service parameter lists (local - extended private)</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>TCBT</td>
<td>TCB table</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>TDBC</td>
<td>Tracking database control blocks</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>TDCB</td>
<td>Tracking log data set DCBS</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>TIB</td>
<td>TIB pool below 16 MB line for LU 6.2</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>TPIP</td>
<td>TPIPE pool above 16 MB line for OTMA</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>TT24</td>
<td>Trace table (24-bit storage)</td>
<td>231</td>
<td>Below</td>
</tr>
<tr>
<td>TTAB</td>
<td>Trace table (31-bit storage)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Subpool</td>
<td>Above or Below the line</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------</td>
<td>---------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>USMU</td>
<td>SMU security update table entry</td>
<td>229</td>
<td>Below</td>
</tr>
<tr>
<td>USRD</td>
<td>User descriptor control blocks</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>VRPL</td>
<td>VSAM RPL/save area stack</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>VTCB</td>
<td>VTAM session control block</td>
<td>251</td>
<td>Above</td>
</tr>
<tr>
<td>VWA</td>
<td>Volatile work area</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>X124</td>
<td>DL/I pool below 16 MB line for MVS/ESA</td>
<td>241</td>
<td>Below</td>
</tr>
<tr>
<td>XMCI</td>
<td>Cross memory ITASK block</td>
<td>241</td>
<td>Above</td>
</tr>
<tr>
<td>XPST</td>
<td>Extended partition specification table (PST) area (dependent region)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>MAIN</td>
<td>Main buffer pool for non-DLI/SAS systems</td>
<td>251</td>
<td></td>
</tr>
<tr>
<td>MAIN</td>
<td>Main buffer pool for DLI/SAS systems</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>PSBW</td>
<td>PSBW buffer pool for non-DLI/SAS systems</td>
<td>251</td>
<td></td>
</tr>
<tr>
<td>PSBW</td>
<td>PSBW buffer pool for DLI/SAS systems</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>DPSB</td>
<td>DPSB buffer pool for non-DLI/SAS systems</td>
<td>251</td>
<td></td>
</tr>
<tr>
<td>DPSB</td>
<td>DPSB buffer pool for DLI/SAS systems</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>DBWP</td>
<td>DBWP buffer pool for non-DLI/SAS systems</td>
<td>251</td>
<td></td>
</tr>
<tr>
<td>DBWP</td>
<td>DBWP buffer pool for DLI/SAS systems</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>CESS</td>
<td>CESS pool</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>CIOP</td>
<td>Communications I/O pool</td>
<td>0</td>
<td>Below</td>
</tr>
<tr>
<td>EMHB</td>
<td>Expedited message handler buffer pool</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>EPCB</td>
<td>Extended PCB pool (Fast Path)</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>FPWP</td>
<td>Fast Path work pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>HIOP</td>
<td>High I/O pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>LUMC</td>
<td>LU 6.2 manager common pool</td>
<td>231</td>
<td>Above</td>
</tr>
<tr>
<td>LUMP</td>
<td>LU 6.2 manager private pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>SPAP</td>
<td>Scratch pad area pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>FPWP</td>
<td>Fast Path work pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>MFBP</td>
<td>MFS buffer pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>QBUF</td>
<td>Message queue buffers</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>QBFL</td>
<td>Queue manager buffer pool</td>
<td>0</td>
<td>Above</td>
</tr>
<tr>
<td>AOIP</td>
<td>Automated operator interface pool</td>
<td>0</td>
<td>Above</td>
</tr>
</tbody>
</table>
8.5.3 Save Area Prefixes

Each active communications ITASK requires a set of register save areas with a save area prefix (SAP). These SAPs are called Dynamic SAPs, because they are dynamically allocated to communications ITASKs as required. The number of dynamic SAPs is fixed (user defined). The IMS monitor Reports report indicates whether a shortage of dynamic save area prefixes has occurred. (See Figure 94 on page 290.)

The Report report indicates the number of times during the IMS monitor interval that IMS had to delay a terminal communication task because no save area prefix was available. It is best if the number is zero or close to zero. If it is not, specify additional SAPs using the SAV= parameter in the control region procedure. If there is always a significant number in this field, investigate why the number is not zero. The IMSPARS internal resource usage reports include two pages on dynamic SAP statistics that may prove useful (see Figure 52 on page 150).
### IMS INTERNAL RESOURCE USAGE

**Dynamic SAP Statistics**

<table>
<thead>
<tr>
<th>TCB NAME</th>
<th>Count / Transact</th>
<th>Count / Second</th>
<th>Interval: 23.38 (HH:MM:SS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOT # AVAIL NON-PRIV SAPS</td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOT # AVAIL PRIV SAPS</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td># SAPS ON STAGE QUEUE</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HI # SAPS ASGN CUR CTRL INTVL</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENT SAP GENERATION</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINIMUM # SAPS</td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENERATION SIZE</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td># EXPANDS DONE</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># CONTRACTIONS DONE</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># MOVES STAGE TO FREE</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># TIMES UPPER LIMIT ON SAPS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># TIMES IN SD FOR SAPS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># TIMES EXPANSION FAILED</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL NON-PRIV SD WAITS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH NON-PRIV SD WAITERS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL PRIV SD WAITS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH PRIV SD WAITERS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td># PRIV DISPS DURING SD</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HI # SAPS ASGN CUR CHKPT INT</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HI # PRIV ASGN CUR CHKPT INT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 52. IMSPARS Internal Resource Usage (Dynamic SAPs)*
8.6 IMS System Checkpoints

From a storage and logging point of view, IMS system checkpoints are periods of intense activity.

8.6.1 Checkpoint Logging

At any IMS system checkpoint, IMS writes out a considerable amount of information to the log. It uses techniques that minimize the amount it writes. For example, it logs only those messages that have been on the queues since the previous checkpoint. Similarly, it logs only SPAs that have remained unchanged since the previous checkpoint.

8.6.1.1 Logging VTCBs

IMS also logs key information from every terminal control block, which has several implications:

- Use ETO. This means that only terminals that actually get used need to be checkpointed.
- Use disabled reference storage for the VTCBs, unless the system does virtually zero paging. Therefore, if VTCBs (VTAM terminal control blocks) get paged out of main storage, they move to expanded storage but never to DASD. This is explained in 8.7.2, “Page Fixing IMS Control Blocks” on page 153.
- Remove from the IMS stage 1 any static terminal definitions for terminals that are never used. This saves storage and speeds up checkpoints.

To expedite the process of getting all VTCBs in main memory, IMS uses the MVS page load facility at the beginning of checkpoint processing and specifies the required address ranges for the VTCBs.

8.6.1.2 SLUP/FINANCE Terminals and the FPACK Parameter

At system checkpoint time, IMS logs all output messages currently in process to make them recoverable. This applies to both full function messages and Fast Path EMH messages. Normally, just a handful of terminals have output in process at checkpoint time. The one exception is with Fast Path message processing for SLUP or FINANCE terminals defined with the FPACK option.

The TERMINAL macro or the ETO logon descriptor for these types of terminal allows a choice of message acknowledgment technique:

```
OPTION=(.,FPACK|NFPACK,.)
```

With NFPACK, when IMS sends the reply message, it requests a definite response. The SLUP device normally responds by sending a DR2 (definite response Type 2) to IMS. On receipt of this, IMS dequeues the output message, so that this reply is no longer in process. A checkpoint after this time does not need to log the EMH buffer.

With FPACK (the default), IMS sends a reply that requests exception response only. The next input message (or an SNA ready-to-receive indicator) is treated as the acknowledgment of the last output message.

The consequence is that nearly all FPACK terminals are in the output in process state (the exceptions are the few that have an input message in process).
checkpoint time, IMS must log nearly all the EMH buffers to make the output messages recoverable.

The impact is significant. Consider a customer with 15,000 SLUP terminals defined with FPACK. Assume an EMH buffer size of 1 KB. At checkpoint, IMS must log about $15,000 \times 1\text{ KB}$, or 15 MBs of data. At 3 MB/second, this takes 5 seconds.

To minimize the number of EMH buffers that need to be logged, it is recommended that the standard SNA definite response protocol be used. In other words, use NF PACK for SLUP and FINANCE terminals. The additional network traffic arising from this recommendation is probably negligible on today's networks.

### 8.6.2 Monitoring Checkpoints

Mention has been made (see 9.20.1, “Monitoring Logging Performance” on page 201) to the IMSPARS logger statistics report, which indicates the number of buffer waits at checkpoint time.

The number, average duration, and maximum duration of system checkpoints are reported in the IMS monitor region summary report (between the “Idle for Intent” and the “Region Occupancy” sections).

**Note**

The IMS monitor does not consider waits for log buffers as IWAITs.

If any of the checkpoints are SNAPQ, then any additional IWAITs resulting from the SNAPQ (for example, reading the queue data sets) are separately reported in the IMS monitor general reports.

Figure 53 shows an example of both reports.

<table>
<thead>
<tr>
<th>IMS MONITOR  <strong><strong>REGION SUMMARY</strong></strong></th>
<th>TRACE START 1996 108 11:00:02</th>
<th>TRACE STOP 1996 108 11:03:00 PAGE 0021</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCCURRENCES</td>
<td>TOTAL</td>
<td>MEAN</td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2860365</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMS MONITOR  ** GENERAL REPORTS **</th>
<th>TRACE START 1996 108 11:00:02</th>
<th>TRACE STOP 1996 108 11:03:00 PAGE 0015</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL IWAIT TIME EVENTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVENT</td>
<td>OCCURRENCES</td>
<td>TOTAL</td>
</tr>
<tr>
<td>IWAITS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMGR SNAPQ CHECK</td>
<td>1</td>
<td>10641</td>
</tr>
</tbody>
</table>

**Figure 53. IMS Monitor System Checkpoint Reports**
8.6.3 Monitoring System Paging at Checkpoint Time
A very common reason for long checkpoint times is that IMS control blocks get paged out between checkpoints and need to be paged in again. This may not show up clearly in the RMF summary reports, but it is apparent in the RMF monitor II SPAG report (see Figure 17 on page 57). This shows paging in each one second time period.

8.7 Page Fixing IMS Pools and Control Blocks
We are summarized the performance related IMS control blocks and buffer pools in the following sections.

8.7.1 Page Fixing Buffer Pools
The pool-related storage that must be page fixed in an IMS system is as follows:
- OSAM database buffer prefixes (DFSVSMxx IOBF=(length,number,N,Y))
- VSAM database IOBs (DFSVSMxx VSAMFIX=(IOB))
- PSB pool in CSA (DLMP)
- PSB pool in DL/I address space (DPSB).

If real storage is a system constraint, take care to minimize the size of each fixed pool. In most cases, I/O to the pool impacts performance less than page faulting through the pool. A smaller fixed pool is better for IMS performance than a larger pageable pool if IMS is executing in a storage-constrained system and the transaction rate is low, thereby causing page faults in the pools.

The new dynamic pool management no longer allows the SPAP, CIOP, or EMHB pools to be page fixed. For these pools, the MVS page load facility is used to bring buffers into real storage.

8.7.2 Page Fixing IMS Control Blocks
We recommend that you page-fix control blocks associated with buffer pools. In particular, the fixing of VSAM and OSAM buffer pool control blocks is specified in the DFSVSMxx member of PROCLIB.

Also, as mentioned in 8.6, “IMS System Checkpoints” on page 151, for an IMS system in a steady state of message processing, the main demands on storage come at system checkpoint times. At this time in particular, IMS needs to access all the terminal control blocks (VTCBs), including those that may not have been used during the last period, and which have consequently been paged out of main memory.

If this is a potential problem, it is recommended that you define VTCBs to use disabled reference storage. This results in the ability of the VTCBs to page out of main memory into expanded storage, but not to DASD. Request this in the DFSDRFxx member (used with DFSFIXxx) by specifying BLOCKS=VTCB.

**Note**

If either DFSFIXxx or DFSDRFxx is used, then both must be specified. If one is not needed, leave the blank control blank. Both members use the same suffix.
The *IMS Installation Guide: Volume 2 (SC26-8024)* lists the control blocks that can be page-fixed using the DFSFIxxx member.

If you use Fast Path, include EPSTs in your fix list. (Failure to do so can cause system abends.)

If real storage is a system constraint, take care to minimize the number of control blocks fixed. In most cases, except for those that are terminal-related, blocks are in the CSA, and because MVS protects CSA differently from the way it protects the private area, these control blocks are not required to be page fixed. However, those control blocks may have to be page fixed if an environment is real-storage constrained, paging at a rate greater than 8 pages per second in the common area of MVS (CSA), or if the control region is paging more than 4 pages per second. The reason for fixing these blocks is that the control region references these blocks while scheduling or terminating a PST, and a page fault suspends the control region. If the control region is suspended, other ITASKs in the control region waiting to be dispatched must wait for the control region to be redispached. In these blocks, page fault in a dependent region does not have the dramatic effect that it has in the control region. The blocks can be fixed by specifying them on the BLOCKS= parameter in the DFSFIxxx member in IMSESA.PROCLIB.

To help you determine the size of the blocks being fixed, the MVS console log and the IMS SYSOUT log from the startup of the control region show the size and location of each block fixed.

---

**8.8 IMS CSA Virtual Storage Considerations**

This section describes the LSO= parameters and the effect they have on CSA usage. For a more detailed description of the use of the CSA in an IMS environment, refer to the *IMS System Administration Guide: System (SC26-8013)*.

**8.8.1 LSO=Y Option**

The local storage option (LSO=Y) moved storage from the MVS common area into the control region address space. The log buffers, DL/I code, OSAM buffer pool, PI ENQ/DEQ storage, and the VSAM buffer pool are in the private area of the IMS control region unless VSAMPL=GLBL is specified in the VSAM OPTIONS statement.

If Fast Path is defined in the IMS system, the PI ENQ/DEQ storage and the log buffers remain in the CSA. The other pools and control blocks remain in the private area of the IMS control region.

If IRLM is selected, ENQ/DEQ storage is not required because IRLM controls the program isolation control block storage, which resides in the private storage area of the IRLM address space.

The VSAM data set control blocks that are created when the data sets are opened are in the CSA if the VSAMPL=GLBL option is selected in the IMS options statement. These blocks are approximately 2.5 KB for an ESDS and 3.5 KB for a KSDS.

We recommend that any randomizers, compression routines, and secondary index maintenance routines reside in the private area of the IMS control region. There is
no valid reason for the routines and working storage to be in the CSA, as the processing of the database calls occurs in the private area of the control region.

8.8.2 LSO=S Option

For local storage option (LSO=S), IMS uses CSA storage for the common OSAM code that is referenced by both the control region and the DL/I separate address space.

IMS also uses CSA storage for the IMS log buffers. The PSB and resident PSB pools in CSA contain the IOPCBs and the Fast Path PCBs. The full-function PCBs move to the DPSB pool in the DL/I separate address space. The size of this pool is about 80% to 90% of the old PSB pool. The part of the PSB pool that remains in CSA is about 10% to 20% of the old pool.

The database buffer pools, OSAM and VSAM, reside in the DL/I separate address space unless VSAMPL=GLBL is specified. The database control blocks that are associated with the open data sets are also moved to the DL/I separate address space.

The resident intent list is moved to the CSA when the LSO=S option is selected. These intent lists can range from 50 KB to more than 400 KB if a test system is defined with a large number of databases and segments.

8.8.3 Virtual Storage Tuning Tips

Define a size for the PSB pool that can hold all the PSBs for the active dependent regions. Determine the size by monitoring the IMS system with the IMS monitor and analyzing the loads for the PSBs and the IWAITs for pool space. IWAITs for pool space are not expected. If any are incurred, they should be for very large PSBs. If a few large PSBs are referenced frequently, make them resident if SCHDTYP=PARALLEL is not specified for the PSB at system definition time.

Review the MVS ICF catalogs to determine whether the CI sizes are too large or too many buffers are allocated for the data and indexes specified. If non-ICF catalogs are still being used, migrate them to ICF catalogs to save common virtual storage.

Another area to investigate is the MVS SQA area of virtual storage. Do not overallocate this area in an IMS environment. The size of this area is determined at IPL. Set the size so it expands a few pages into the CSA area. This assures that no space is wasted in the SQA area of common storage. Another area of virtual storage tuning can come from the analysis of the pageable link pack area in MVS. Focus your analysis on products or modules that are not used in the IMS system but are there to provide compatibility with a TSO or batch system. If this is the situation and virtual storage is a constraint, delete the modules from the PLPA or MLPA.

Review the VTAM buffers to determine whether they are overallocated. If they are too large, reduce them in order to reduce the usage of virtual storage. Allow the VTAM buffers to go into expansion at startup and shutdown. Then allow them to go into expansion only during peak IMS processing and only in the exceptional situation.
8.8.4 Running Multiple IMS Systems in One MVS

In many organizations, it is necessary to run a large number of IMS systems concurrently in an application development environment. This is often the case where several application teams are working in parallel, and each team requires a separate IMS system.

The following outlines an approach to maximizing the number of IMS systems that can run in a single IMS. It is likely to be necessary for only those organizations that require a very large number of IMSs loaded into a single MVS. With IMS 5.1, some customers are running ten IMSs concurrently, and it is possible to run 13 or more in a single MVS/ESA system. Take the following steps:

1. Identify how many IMS systems are actually required.
   
   Look at the number of IMS systems you currently have, and determine whether they all need to be run separately and concurrently. Often, where the requirements of several application groups do not conflict, it is possible to merge several separate IMS systems into one.

2. Determine the number you need to run concurrently.
   
   Having determined which IMS systems you need to keep, identify which IMSs need to be available all the time. It may be possible to overlap the execution of some of these IMSs, so that the number active at any point in time is reduced. Then see which of the other IMSs could be started independently.

3. Start an IMS system.
   
   From the list, choose the IMS system that uses the most IMS features. The object of the exercise is to load as many of the IMS modules as possible into memory in a single execution, so you can identify which modules are used in your organization.

4. Identify the areas of CSA used by this IMS.
   
   IMS uses CSA for many different things. You need to identify which areas of CSA are used for IMS code, and the modules the code is from. Often an eyecatcher label near the top of the area identifies the module name.

5. Determine which of these areas are code modules.
   
   Blocks of memory in CSA have an eyecatcher that is used to identify the corresponding code module.

6. Identify the code modules that can be reentered or reusable (RENT/REUS).
   
   Any modules that are reentrant and reusable do not modify themselves while executing. Consequently, they need to be loaded into memory only once and can be shared by all running IMS systems.

7. Load these modules into MLPA.
   
   The modules can be listed on one of the SYS1.PARMLIB members, so MVS loads the module into MLPA. MVS then uses this copy of the module, rather than loading it into CSA for each running IMS.

8. Use the modules preloaded into MLPA.
   
   Alter your program-load parameters to look at MLPA before CSA or any job or step libraries.
Advantages: Although each IMS needs, in total, the same amount of memory as before, the aggregate of all the IMSs is less as a result of the sharing of the common modules. Only one copy of each module is required for the MVS system, rather than one copy for each IMS.

You can run more IMSs in parallel in the same MVS.

Disadvantages: It is necessary to repeat the initial program load (IPL) of the system whenever you need to use a different copy of any of these modules. They are loaded only at MVS initialization.

The list of modules loaded into CSA below the line changes with each new release of IMS. For each of the last five releases of IMS, the amount of storage being used in CSA below the line has been reduced. It is necessary to validate the list again with each new release of IMS.

The following Table 12 contains a list of modules to consider for loading into MLPA. Please note that this list is only a good start, and check each module to make sure it is still RENT/REUS. These attributes certainly change with IMS release levels, and may change with a PTF.

<table>
<thead>
<tr>
<th>Fast Path</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DBFDBAU0</td>
<td>DBFDEDB0</td>
<td>DBFEMH00</td>
</tr>
<tr>
<td>DBFFORI0</td>
<td>DBFHDC40</td>
<td>DBFIRC10</td>
</tr>
<tr>
<td>DBFLRH00</td>
<td>DBFMMO0</td>
<td>DBFMSDB0</td>
</tr>
<tr>
<td>DBFSYNC0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Full Function</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DFSABND0</td>
<td>DFSAOSF0</td>
<td>DFSAOS10</td>
</tr>
<tr>
<td>DFSAOS60</td>
<td>DFSAOS70</td>
<td>DFSAOS80</td>
</tr>
<tr>
<td>DFSASK00</td>
<td>DFSBCK00</td>
<td>DFSBC000</td>
</tr>
<tr>
<td>DFSBML00</td>
<td>DFSCPY00</td>
<td>DFSCWU00</td>
</tr>
<tr>
<td>DFSDBDR0</td>
<td>DFSDBIE0</td>
<td>DFSDCAP0</td>
</tr>
<tr>
<td>DFSDLA00</td>
<td>DFSDLA50</td>
<td>DFSDLTR0</td>
</tr>
<tr>
<td>DFSDRCL0</td>
<td>DFSDRID0</td>
<td>DFSESD30</td>
</tr>
<tr>
<td>DFSESGL0</td>
<td>DFSESMG0</td>
<td>DFSFDLF0</td>
</tr>
<tr>
<td>DFSFLAT0</td>
<td>DFSFLLG0</td>
<td>DFSFLRC0</td>
</tr>
<tr>
<td>DFSFTIM0</td>
<td>DFSFUNL0</td>
<td>DFSHCMS0</td>
</tr>
<tr>
<td>DFSHDAI0</td>
<td>DFSHRDB0</td>
<td>DFSHSRV0</td>
</tr>
<tr>
<td>DFSISIS0</td>
<td>DFSISIS0</td>
<td>DFSLATE0</td>
</tr>
<tr>
<td>DFSLAWE0</td>
<td>DFSMDA10</td>
<td>DFSMNTR0</td>
</tr>
<tr>
<td>DFSMODU0</td>
<td>DFSMPOS0</td>
<td>DFSMRTR0</td>
</tr>
<tr>
<td>DFSPCRP0</td>
<td>DFSPLG0</td>
<td>DFSRO010</td>
</tr>
<tr>
<td>DFSRCQM0</td>
<td>DFSRCQR0</td>
<td>DFSRELP0</td>
</tr>
<tr>
<td>DFSRTM00</td>
<td>DFSSSI10</td>
<td>DFSSTB10</td>
</tr>
<tr>
<td>DFSCHR0</td>
<td>DFSDDL0</td>
<td>DFSDDL80</td>
</tr>
</tbody>
</table>

Table 12 (Page 1 of 2). List of Modules That Can Be Loaded into MLPA with IMS 5.1
8.9 Storage Isolation

Storage isolation allows you to specify a minimum number of real storage frames to the IMS control region and DL/I separate address space. The IMS systems that require storage isolation are those that either have a low transaction arrival rate or execute in a real-storage-constrained environment. You can implement storage isolation in an IMS system even if the system is not in the constrained environment.

Storage isolation does not cause a degradation of the system if it is implemented, and it can help response time if the load on the MVS system increases without warning. Storage isolation allows you, through the use of MVS IPS, to specify a minimum number of real frames to be allocated to the IMS control region and the DL/I separate address space in order to achieve response-time objectives.

The dependent regions can also be included in the specification of the storage isolation parameters, if required. These dependent regions could be a WFI or a never-ending BMP.

At the same time, a minimum and maximum working set size can be specified for a test IMS system in order to reduce large spikes in paging when the test system becomes active.

8.9.1 Reports Required for Storage-Isolation Implementation

RMF Monitor II reports are needed to supply data to create the IPS member with the storage isolation parameters. These reports are the ASD, ARD, SRCS, and the SPAG; they show the working set size of each address space, the page fault rate for each address space, the working set size of the common area, and where the paging is occurring.

8.9.2 Defining the Storage-Isolation Working Set

The target working set size of the private address space is defined by performance group number. A minimum and a maximum size can be specified. The minimum size for the IMS control region and DL/I separate address space needs to be large enough so that the paging rate for the IMS private area averages about 1 to 2 page faults per second. This page fault rate is the NVRT column on the ARD report. Specify the maximum size for the IMS regions as "*" so if IMS needs the frames, they will be available. The "*" means that there is no maximum number of frames. The working-set sizes include all private frames assigned to the address space.
Included are the LSQA frames, fixed private frames, and the pageable private frames.

Define the IMS control region and DL/I separate address space as two unique performance groups, in order to specify different working-set sizes for each region. Specify only one performance period for each performance group.

We recommend that IMS address spaces do not incur more than one to two page faults per second. This rate is a guideline where the page resolution service time is approximately 80 ms or less. If the page resolution time is different, adjust the page fault rate either up or down. The rate includes both the common and private areas. The page resolution time is critical to IMS response time; therefore, monitor the page fault rate and adjust it in conjunction with the page resolution time.

8.9.3 Paging Rate for Storage Isolation
The paging rate in the IPS for the private address space is specified per CPU second for non-cross-memory address spaces and per elapsed second for cross-memory address spaces and address spaces that use the PPGRTR parameter. This parameter can be used for any address space and is recommended for the IMS control region, the DL/I address space, and the IRLM address spaces. The IMS control region is considered a cross-memory address space if LSO=Y is specified at execution time. If the LSO=S option is selected, the DL/I separate address space is the cross-memory address space, instead of the control region.

8.9.4 Common Area Storage Isolation
The common-area target working set for storage isolation includes the PLPA, MLPA, and CSA pageable frames only. It does not include any fixed frames in PLPA and CSA. Make the size of the minimum working set for the common area large enough so the common area does not page more than 6 to 8 pages per second.

The paging rate can go higher or lower depending on the page resolution time and the number of address spaces that are active and taking page faults. The paging rate specification in the IPS for the common area is per elapsed second and not per CPU second. This differs from the specification for the private non-cross-memory address spaces.

The working set size (WSS) does not go below the minimum WSS minus a few frames unless there is a critical storage shortage. If there is a critical storage shortage, the SRM steals the two oldest frames from any address space, regardless of storage isolation specifications. The system goes into a critical storage shortage when all address spaces are at a minimum number of allocated frames and the system needs more frames.

8.9.5 Dependent Region Use of Storage Isolation
If a message region is a WFI or BMP, it can also be included in the storage isolation parameters as it will be the only program to execute in that region, and the working set size should be fairly small.
8.9.6 Monitoring the Storage Isolation Specification

Do not overspecify the private working set size, as overprotecting private storage tends to drive the paging rate up. Frames that are protected are not stolen.

Once the IPS is set up to activate the storage isolation parameters, monitor the system with RMF monitor in real time in order to make any adjustments to the parameters.

If not monitored, and if the parameters are specified on the high side (overprotection), the system could degrade to a very poor performance level. Also, if not monitored and if the IMS working set profile change to require more storage, IMS could page heavily. This happens if buffer pool sizes are changed or if a new system is defined and the applications that were added become active.
Chapter 9. Understanding IMS Buffer Pool Storage

The IMS transaction I/O profile is largely affected by the size of the IMS buffer pools and blocks. The objective in tuning the buffer pools is to make them large enough to minimize the I/O activity in the pools, without increasing the system paging rate.

9.1 IMS Buffer Pool Storage

Figure 54 identifies IMS storage pools problems.

MAIN STORAGE RESOURCE -- IMS POOLS

1. If the transaction rate is less than 2 per second, the IMS pools are paging, or both, review pool page fixing recommendations.

2. If an I/O rate of 0.5 or more per transaction to the MSGQ pool or 15 I/Os per second exists, analyze record size and number of buffers.

3. If more than 1 format I/O per transaction, analyze size and use of the MFS pool.

4. If more than 1 I/O per schedule to ACBLIB or 12 I/Os per second, check DMB and PSB pool utilization.

5. If the number of database I/Os is greater than one-half the number of DL/I calls, check database buffer pools or analyze the DL/I Call Summary Report.

6. No IWAITs for CIOP, HIOP, SPAP pools should occur.

7. Evaluate the Exclusive Control Enqueue/Dequeue work area storage.

8. Return.
The IMS transaction I/O profile is largely affected by the size of the IMS buffer pools and blocks. The objective in tuning the buffer pools is to make them large enough to minimize the I/O activity in the pools, without increasing the system paging rate.

The IMS pools are monitored with the IMS monitor and the /DIS POOL command. The sections that follow describe the critical pools and the key factors to consider when determining the optimum size of each.

IMSPARS provides excellent reports on IMS storage pools and is very helpful for storage management analysis. This is especially true with the new dynamically managed pools.

### 9.2 Dynamic Pool Management

The following pools are dynamic:

- **CIOP**  Communication I/O pool
- **HIOP**  High I/O pool
- **SPAP**  Scratch pad area pool (formerly CWAP)
- **CESS**  Communication external subsystem
- **AOIP**  Automated operator (Type 2) pool
- **EMHB**  Expedited message handler buffers
- **FPWP**  Fast-Path work pool
- **LUMP**  LU 6.2 manager private buffer pool
- **LUMC**  LU 6.2 manager common buffer pool

#### 9.2.1 Dynamic Pool Manager (DFSPool)

IMS has several pool and storage managers. The dynamic pool manager (DFSPool) was introduced in IM/ESA Version 4.1. Prior to DFSPool, pool managers typically allocated an amount of storage that exactly matched the requested amount. With this technique, it is necessary to keep track of many different sized pieces of allocated and free storage. Although this works well for small pools, it becomes inefficient with large pools. This method also leads to problems of storage fragmentation. The other problem typical of the older storage managers is that a pool is allocated at IMS initialization and remains static in size thereafter.

The characteristics of DFSPool are:

- The size of each pool can change dynamically according to the system demands.
- Each pool consists of multiple blocks of fixed-length buffers.
- Fixed-length buffers are used to satisfy variable length requests.
- Bit maps are used to manage storage.
9.2.2 The DFSPOOL Pool Structure

Each fixed pool (see Figure 55) consists of zero or more noncontiguous blocks of storage anchored off a pool header. By obtaining new blocks and releasing unused blocks, a pool can be expanded and contracted as needed during the execution of IMS. Each block is divided into a number of fixed-length buffers. The size and number of buffers each block contains may vary from block to block within a pool.

![DFSPOOL Structure Diagram](image)

Each pool is allocated with a maximum of 32 different buffer sizes, though typically only 8 are used. The pool header contains a noncompressible block chain anchor and a compressible block chain anchor for each buffer size available. The buffer sizes and the block chain anchors are placed in the pool header in ascending order by buffer size. The pool header also contains an oversized block chain anchor.

If the requested size is larger than the largest buffer size available, a block containing a single buffer of the requested size is obtained. Blocks obtained in this manner are placed on the oversized chain. The intention of the oversized chain is to allow for exceptional requests, because normal processing usually does not need any oversized buffers.

A block header is allocated for each block. The header contains information on the block such as the length of the block, the address of the first buffer within the block, the size and number of buffers within the block, and so on.
9.2.3 Primary and Secondary Blocks

A pool can contain, for each buffer size, one primary block of buffers and any number of secondary blocks (Figure 56). The primary block can be obtained at IMS startup (user option), in which case it is never compressed or deleted. Alternatively, if the primary block is defined as compressible, it is allocated only when actually needed and is eligible for deletion when no longer required.

Secondary blocks are always eligible for compression.

![Diagram of DFSPOOL Primary and Secondary Blocks](image)

Figure 56. DFSPOOL Primary and Secondary Blocks

The number of buffers in the primary block may differ from that in the secondary blocks for the same buffer size. Blocks obtained during initialization cannot be compressed; therefore, no latches are required when allocating or releasing a buffer from one of these blocks. Noncompressible blocks are anchored off the pool header but are not placed on the compressible block chain.

If the primary block is not obtained until the first GET request, it is placed on the compressible block chain along with any secondary blocks. It is also anchored at the pool header. The pool latch must be held in shared mode before the blocks on the compressible chains can be scanned.
9.2.4 Bit Maps

The buffers within a block are managed with a bit map (Figure 57 ) residing in the block header. The bit map contains a bit for every buffer within the block. Each bit is initialized to zero. When a buffer is allocated, its corresponding bit is set to one and remains so until the buffer is deallocated.

9.2.5 Buffer Allocation Algorithm

When a requestor needs buffer space, it asks DFSPOOL, specifying the buffer size required. DFSPOOL uses the following to find a suitable buffer.

The buffer size used to satisfy an incoming request is determined on a best fit basis. Unless the size of the buffer requested is the same size as the actual buffer, there will be some unused storage between what the caller views as the end of the buffer and the actual end of the buffer. The buffer the user receives appears to be of the size requested. Any unused space is transparent.

The buffer sizes available are kept in the pool header in ascending order. When a buffer request is received, the storage manager scans the pool header, starting with the smallest buffer size, until it finds a buffer size large enough to satisfy the request. The noncompressible block, if there is one, is examined first. If there are no free buffers in the noncompressible block, or the block does not exist, the compressible block chain is examined. Before scanning the chain, the pool latch is obtained in shared mode. The storage manager scans the chain of blocks until
either a free buffer is found or the end of the chain is reached. Once the buffer is allocated, the pool latch (if held) is released. The buffer address is calculated and returned to the caller.

If no free buffers of the needed size are available, a new block is obtained. If the primary block has not been obtained, the primary block buffer count is used to calculate the block size; otherwise the secondary block buffer count is used. Once the block has been initialized, the first buffer in the new block is allocated. The block is added to the end of the chain, prior to releasing the storage pool latch. Once the latch is released, the buffer address is calculated and returned to the caller.

If the request size is larger than the maximum buffer size, the oversized chain is scanned for the first available buffer large enough to satisfy the request. If an oversized buffer is not available, a block is obtained containing a single buffer of the requested size. The block is placed at the head of the oversized chain. Each buffer on the oversized chain has an associated block header and appears to the storage manager as a block containing a single buffer. The buffer is managed with a bit map contained in the block header. All oversized buffers that are unused at the next compression interval are released.

9.2.6 Buffer Release

When the user of a buffer has finished with it, the buffer is released by the requestor. Because the requestor knows only the virtual storage address of the buffer, the storage manager must first find the corresponding block, and then update the bit map in the block header. The scanning begins with the noncompressible blocks, if any. The buffer address is compared with the block address. If the buffer address is greater than the beginning address of the block, the buffer address is compared with the last address in the block to determine if the buffer resides within the block. If the buffer address is less than the block address, the next noncompressible block is examined.

If the buffer is not found in a noncompressible block, the pool latch is obtained in shared mode and the scanning continues with the compressible block chains. If the buffer does not reside in a block, the address of the next block on the chain is obtained from the block header and the next block is examined. If no more blocks are on the chain, the block chain anchor for the next buffer size is obtained from the pool header and the process repeats.

The scanning continues until the block containing the buffer is found or until every block has been examined, including any oversized blocks. If the block containing the buffer is not found, an invalid return code is returned to the caller.

9.2.7 Pool Expansion and Upper Limit

As explained, when a buffer is requested but none is available, the pool is expanded by adding a new block. An upper expansion limit for each DFSPOOL-managed pool may be specified using EXEC parameters (except for CESS pool). Message DFS0637W is issued if a pool reaches the upper expansion limit. However, it is generally recommended to let the upper limits default to 2 GB - 1.

You may feel uncomfortable with the possibility of running out of storage as a result of expanding a pool until all the CSA or private storage in the system is allocated.
If IMS storage pools residing in CSA were to expand uncontrolled, then IMS could use all the available storage in the system and cause MVS to abend. To eliminate this concern, use the EXEC parameters to specify maximum sizes significantly larger than you expect to need in normal running.

The upper expansion limit is used when a new block must be obtained. The size of the new block is added to the current pool size. If the resulting size is larger than the upper limit, the storage is not allocated. If requested by the caller, the storage manager waits for storage to become available. If the storage request is not satisfied after two compression cycles, a nonzero return code is returned to the caller.

If the caller did not request the storage manager to wait, a non-zero return code is returned immediately.

The DFS0637W message is issued during IMS checkpoints for each pool that reached its upper expansion limit since the last checkpoint.

### 9.2.8 Pool Compression

Storage pool compression is an asynchronous two-pass process that occurs at intervals of not less than 2 minutes. On the first pass, the storage manager scans the compressible block chains for each pool and determines how many, if any, blocks can be deleted from each chain. A block is eligible for deletion if the number of allocated buffers is zero. No compression is actually done in this pass; thus no latches need to be held.

If there is no compression to be done, the timer is reset and the compression routine terminates. If there are blocks to compress, the storage manager reexamines each pool to determine which blocks can be deleted. An exclusive pool latch is obtained before any compression is done. All blocks to be deleted are removed from the block chains and added to a compression chain. Once a pool is examined and all blocks being deleted are removed from their chains and added to the compression chain, the pool latch is released. The storage manager subsequently frees all of the blocks on the compression chain. This process is repeated for each pool.

With this technique, the pool latch is held exclusively for the absolute minimum length of time—just the time to manipulate a few pointers on a chain. Actual deletion is done asynchronously. Compression is the only function that requires that the pool latch be held exclusively. All other processing of the compressible blocks acquires the latch in shared mode.

To prevent thrashing, not all eligible blocks are deleted in one compression interval. If a block chain contains more than one eligible block, only one-half of the eligible blocks are deleted. If only one block on a chain is eligible for deletion, the block is not deleted unless at least one block on the chain has remained unused since the last compression interval.
9.2.9 Defining Pool Sizes

Pool definitions are read at IMS initialization time. IMS includes the default pool definitions (see Figure 58) in module DFSSPM10. Its values can be overridden by a user-defined DFSSPMnn PROCLIB member (the suffix is specified with the IMS EXEC parameter, SPM=nn).

Each definition also contains the pool name, pool type, subpool, location of storage, upper limit and whether an 8-byte character string is to be added to the end of each buffer for overlay detection.

The buffer definitions contain the sizes from which the storage manager is allowed to choose when allocating a buffer from the pool. For each buffer size, the
definition specifies how many buffers to obtain in the primary block, how many
buffers to obtain in secondary blocks, and whether to obtain the primary block when
the pool is allocated. If the primary block is obtained during initialization, it is not
released during compression.

The DFSSPM10 load module also contains definitions for the following nondynamic
pools that are allocated by the DFSPOOL storage manager but managed by their
associated functions: MFBP, EPCB, QBUF, QBFL.

The DFSSPMxx PROCLIB member can be used to override the defaults for one or
more pools.

FPL=poolname,(size,pbuf,dbuf,init),(size,pbuf,dbuf,init),(...)

│ │ │ │
│ │ │ └─/SM590000 Y/N, PRIMARY ALLOC DURING INIT
│ │ │
│ │ └──/SM590000 SECONDARY ALLOC, BUFFERS PER BLOCK
│ │
│ └──/SM590000 PRIMARY ALLOC, BUFFERS PER BLOCK
│
└──/SM590000 BUFFER SIZE IN BYTES

For example, to replace buffer definitions for HIOP, you could use

FPL=HIOP,(200,20,30,Y),(300,20,20,N),(400,10,20,N)
FPL=HIOP,(500,10,30,Y),(600,10,10,N),(700,10,20,N)
FPL=HIOP,(800,5,10,Y),(900,5,5,N)

Note that this pool definition extends over three lines. No continuation character is
used in column 72.

A user-defined DFSSPMxx PROCLIB member is not required, but is recommended
to ensure efficient use of storage. During IMS initialization, the member is read and
each of the statements is validated. The default buffer definitions for the specified
pools are replaced with the definitions in the PROCLIB member. The updated pool
definitions contained in DFSSPM10 are then used to allocate the pool.

A maximum of 32 buffer definitions is allowed per pool, but generally 8 are
adequate. The size of buffer used to satisfy a request is determined on a best-fix
basis. If the member contains more than 32 buffer definitions for a single pool, only
the first 32 valid definitions are used. The remaining definitions are ignored.
9.2.10 Storage Usage

Table 13 shows where each of the DFSPOOL pools resides.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Pool Name</th>
<th>Location</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIOP</td>
<td>Communications I/O pool</td>
<td>IMSCTL/PVT</td>
<td>TP buffers</td>
</tr>
<tr>
<td>HIOP</td>
<td>Communications high I/O pool</td>
<td>IMSCTL/EPA</td>
<td>TP buffers</td>
</tr>
<tr>
<td>SPAP</td>
<td>Scratch pad area pool</td>
<td>IMSCTL/EPA</td>
<td>Scratch pad areas</td>
</tr>
<tr>
<td>CESS</td>
<td>Communication external subsystem</td>
<td>ECSA</td>
<td>External subsystem buffers</td>
</tr>
<tr>
<td>AOIP</td>
<td>Automated operator interface pool (Type 2)</td>
<td>IMSCTL/EPA</td>
<td>AOI exit messages</td>
</tr>
<tr>
<td>FPWP</td>
<td>Fast-Path work pool</td>
<td>IMSCTL/EPA</td>
<td>Fast-Path work buffers</td>
</tr>
<tr>
<td>EMHB</td>
<td>Expedited message handler buffer</td>
<td>ECSA</td>
<td>EMH buffers</td>
</tr>
<tr>
<td>LUMP</td>
<td>LU6.2 Mgr private</td>
<td>IMSCTL/EPA</td>
<td>LU6.2 TP buffers</td>
</tr>
<tr>
<td>LUMC</td>
<td>LU6.2 Mgr common pool</td>
<td>ECSA</td>
<td>LU6.2 misc. work areas</td>
</tr>
</tbody>
</table>

The pool headers for each of the pools reside in extended CSA storage.

A table is created during initialization that is used during buffer allocation. The table contains 256 two-byte entries. This table also resides in extended CSA.

The load module containing the default pool definitions is also loaded into extended CSA. 16 KB of extended CSA storage is used for the storage manager trace tables.

The dynamic storage management does not allow the SPAP, CIOP, or EMHB pools to be page fixed; the MVS page load facility is used to bring buffers into real storage.

9.2.11 Checkpoint Statistics

X’45’ log records are written at IMS checkpoint time for each of the buffer pools. The new DFSPOOL buffer pools use X’450E’ records, and the original pool manager pools use X’4505’ records.

9.2.12 IMSPARS IRU Reports for DFSPOOL

A sample IMSPARS internal resource usage report for the HIOP is shown in Figure 59 on page 172. From this sample, you can see that the pool size at the last system checkpoint encountered on the log by IMSPARS (Max Pool Size) was over 4 MB. This was also the maximum over the period analyzed by IMSPARS. However, there had been a time previously when the HIOP had grown to over 7 MB (Max Pool Size Since IMS Restart).
Eight buffer sizes are defined. Consider the column for a buffer size of 3,080 bytes:

- The primary block contains 400 buffers of 3080 bytes each.
- Each secondary block contains 100 buffers.
- The maximum number of 3080-byte buffers in use since IMS restart is 699.
- The maximum number of blocks in use since IMS restart is four (the primary and three secondary blocks).
- The primary block is permanently allocated (Initial Alloc = YES).
- The average buffer request, satisfied with 3080-byte buffers, is for 2154 bytes.
- The number of requests satisfied with 3080-byte buffers is 9618.
- The average number of requests per second is 24.

At the end of the analysis period:
- The average number of 3080-byte blocks is one (just the primary).
- No recent pool expansions took place.
- On average, just two 3080-byte buffers were in use at a time.
- The maximum number of buffers in use was ten and the minimum zero.

- There have been no page faults causing IWAITs.

There are 400 buffers permanently allocated of size 4104 bytes (about 1.6 MB in total), but which were not used in the period analyzed. As a final observation, the total number of get buffer requests (1714 per second) indicates a very heavy transaction rate. The overall average buffer count is 32. Clearly then, HIOP buffers must be released very soon after being allocated.
### IMS INTERNAL RESOURCE USAGE

**FIXED POOL USAGE STATISTICS**

**POOL NAME** | **HIOP**
---|---
**CURRENT POOL SIZE (BYTES)** | 4,389,784
**MAX POOL SIZE** | 4,389,784
**CURRENT # BYTES IN OVERSIZE BLOCKS** | 8,288
**MAX POOL SIZE SINCE IMS RESTART** | 7,403,320

<table>
<thead>
<tr>
<th>BUFFER SIZE</th>
<th>224</th>
<th>392</th>
<th>560</th>
<th>976</th>
<th>1,288</th>
<th>1,648</th>
<th>3,080</th>
<th>4,104</th>
<th>OVERSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td># BUFFERS/BLOCK (PRIMARY BLKS)</td>
<td>50</td>
<td>800</td>
<td>500</td>
<td>500</td>
<td>30</td>
<td>200</td>
<td>400</td>
<td>400</td>
<td>0</td>
</tr>
<tr>
<td># BUFFERS/BLK (SECONDARY BLKS)</td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>MAX BUFFER CNT SINCE INIT</td>
<td>19</td>
<td>66</td>
<td>47</td>
<td>2,505</td>
<td>16</td>
<td>87</td>
<td>699</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MAX # BLOCKS SINCE INIT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>22</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>INITIAL ALLOC (YES/NO)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>AVERAGE SIZE REQUESTED</td>
<td>160</td>
<td>310</td>
<td>473</td>
<td>830</td>
<td>1,226</td>
<td>1,643</td>
<td>2,154</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># GET REQS FOR THIS BUFFER SIZE</td>
<td>39,507</td>
<td>165K</td>
<td>50,248</td>
<td>61,839</td>
<td>52,870</td>
<td>319K</td>
<td>9,618</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># OF TIMES UPPER LIMIT REACHED</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># TIMES LARGER BUF SIZE USED</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AVERAGE BLOCK COUNT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>HI BLK CNT SINCE LAST CKPT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LO BLK CNT SINCE LAST CKPT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BLKS ALLOCATED (EXPANSION)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BLKS RELEASED (COMPRESSION)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AVERAGE BUFFER COUNT</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HI BUF CNT SINCE CHECKPOINT</td>
<td>11</td>
<td>66</td>
<td>30</td>
<td>30</td>
<td>15</td>
<td>47</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LO BUF CNT SINCE CHECKPOINT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PAGE LOAD INVOKED, NO IWAIT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PAGE LOAD CAUSED IWAIT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 59. IMSPIARS Internal Resource Usage Report for HIOP*
9.2.13 IMS Dynamic Pool Tuning Recommendations

- Maximize buffers in the primary blocks.
  
The best performance may be achieved from the storage manager by ensuring that all buffers are obtained out of the noncompressible blocks. Therefore, use the initial option for persistent storage. The size can be established after samples are taken at periods of peak traffic.

- Minimize the number of blocks per buffer set.
  
  Avoid the large system effects of scanning long chains of blocks; overallocation can cost cycles and storage.

- Spread the buffer set; distribution saves virtual storage.
  
  A wide distribution of buffer sizes improves the chances of better fits of messages to buffers and thereby avoids wasting virtual storage.

- Do not overconstrain pools.
  
  It is essential that the upper limit for a pool not be too low. If the storage manager makes a request exceeding the limit, results are unpredictable. For the EMHB pool, a message is returned to the terminal; for other pools, an abend of IMS may ensue.

- Avoid oversize traffic.
  
  Use of the oversize buffers is very costly; individual getmains, inefficient matching of large message length to oversize buffer, and compression overheads may follow.

  Use of oversized buffers is shown by the /DIS POOL command (the overflow value is not zero).

Further guidelines are:

- Do not exaggerate the fine tuning; requirements are mostly met by self-adjusting.

- Sample the data to create more efficient buffer set groups.

- Avoid large system effects and pool fragmentation.

9.3 RECANY Pool

The RECANY buffers are used by IMS to receive all VTAM input messages. Whenever a RECANY buffer has finished processing an input message, IMS reissues a VTAM RECEIVE (ANY) call and VTAM uses it for a subsequent input message.

Each RECANY buffer has an ITASK for. Whenever a message is received into the RECANY buffer, the corresponding ITASK touches the appropriate terminal control block (VTCB) using PGLOAD, and if a page in is necessary, the ITASK IWAITs. This avoids the IMS control TCB suffering an MVS wait.

Since VTAM 3.4, if VTAM receives a message for IMS when no RECANYs are outstanding, it uses an MVS/ESA data space to queue input messages pending RECEIVEs being issued by IMS.
9.3.1 RECANY Processing

When VTAM posts a RECANY ITASK—in other words, a RECANY buffer has had data placed in it—IMS normally does the following:

- Determines the CLB (terminal control block within the VTCB)
- Transfers control to a communications ITASK for that terminal
- Processes the buffer (MFS editing) using the HIOP as work space and places the message into a message queue pool buffer or the terminal's EMH buffer
- Reissues the RECANY to VTAM

In other words, the RECANY buffer is held all the way through the processing of the input message.

In a few cases, this might cause all RECANY buffers to be used concurrently, thus causing a VTAM bottleneck. An example is where many transactions are in nonresponse mode and defined at IMSGEN with DCLWA=YES (the default). In this case, the input message is logged (message ENQ, type X ‘35’ log record) with a LOG WAIT WRITE before the RECEIVE is reissued. This can result in all the RECANY buffers being in use, waiting for the log timer to pop and drive physical logging of the input messages.

In most IMS systems, between 16 and 24 RECANY buffers are sufficient. Some systems do require more, and values up to 500 are allowed. There is negligible overhead in requesting more than necessary.

9.3.2 RECANY Monitoring

Information about RECANY buffer usage, including the high-water mark, is logged at each system checkpoint in type X ‘450D’ log records. However, it is normal to have occasions when all RECANY buffers are filled, and having a high water mark at the maximum is not a cause of concern.

Two IMSPARS internal resource usage reports include references to RECANY buffers. See Figure 60 on page 175 for examples of both. The section “Miscellaneous Statistics” reports the current number of receive-any buffers in use, and the maximum number used. The section “Storage Pool Statistics IMSPARS” also reports the total size of the RECANY buffers (though not the number of buffers defined).
## IMS Internal Resource Usage

### Miscellaneous Statistics

**Interval:** 32.42 (HHHH.MM.SS)

<table>
<thead>
<tr>
<th>Count/Second</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest PST Used</td>
<td>32</td>
</tr>
<tr>
<td>Highest CCB ID Used</td>
<td>0</td>
</tr>
<tr>
<td>Current # of CCBS in Use</td>
<td>0</td>
</tr>
<tr>
<td>Number of Conversations Started</td>
<td>0.00</td>
</tr>
<tr>
<td>Number of Conversational Transactions</td>
<td>0.00</td>
</tr>
<tr>
<td>Number of Wait-For-Input Transactions</td>
<td>0.00</td>
</tr>
<tr>
<td>Number of Nonrecoverable Transactions</td>
<td>96.07</td>
</tr>
<tr>
<td>Number of Response-Mode Transactions</td>
<td>25,929</td>
</tr>
<tr>
<td>Maximum # Receive Any Buffer Used</td>
<td>11.28</td>
</tr>
<tr>
<td>IMS Release Level</td>
<td>5.1.0</td>
</tr>
<tr>
<td>Statistic Records Version Level</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 60: IMSPARS Internal Resource Usage Reports for RECANY Buffers
9.4 Communication Pools

This set of pools includes the CIOP, HIOP, SPAP (formerly CWAP), and the two LU 6.2 pools. All are managed by the dynamic pool manager, DFSPOOL.9.2, “Dynamic Pool Management” on page 162 gives information and guidance that is equally applicable to all three pools.

These pools cannot be page fixed. The MVS Page Load Facility is used to bring buffers into real storage.

Additional notes about each individual pool follow.

9.5 Communications I/O Pool

The CIOP pool is used for IMS command work areas and BTAM I/O messages (if any BTAM devices are used with IMS). It is also used to hold the MFS control blocks (MID/MOD, DIF/DOF) when in Test-MFS mode. It is the only dynamic pool stored below the 16 MB line.

9.6 High Communications I/O Pool

A VTAM input message is moved from the RECANY buffer to a HIOP buffer, where it is edited by MFS, ISC edit, or basic edit, as appropriate. It is then moved to a queue buffer or to an EMH buffer (EMHB).

Output messages are moved to the HIOP from the message queue pool or EMHB. If appropriate, MFS editing takes place, and then the VTAM SEND is issued.

9.7 Scratch Pad Area Pool

The SPA pool (SPAP) contains the storage for holding SPAs. This includes the SPAs for both active conversations and held conversations.

9.8 LU 6.2 Pools (LUMC and LUMP)

APPC/IMS requires working storage when processing APPC conversations. In some cases, the storage is required in ECSA, in which the LU manager common pool (LUMC) is used. In other cases, the storage is required in the extended private area of the control region, and the LU manager private pool (LUMP) is used.

Both of these pools are managed by the DFSPOOL storage manager and expand and contract as required. See 9.2.1, “Dynamic Pool Manager (DFSPOOL)” on page 162 for further information about defining and monitoring these pools.

9.9 Message Queue Pool
9.9.1 Message Queue Overview

IMS has three message queue data sets: the short message queue (SHMSG), the long message queue (LGMSG), and the QBLKs data set. Input messages are queued off the scheduler message block (SMB) for the transaction. Output messages are queued off a QBLK, which in turn is chained off an LTERM (the control block is the communications name table or CNT). Each QBLK has a number of subqueues.

IMS messages are composed of one or more segments. The message segments are chained together off the SMB or QBLK.

9.9.2 Rules for Message Segment Use of Logical Records

There are rules for allocating segments to logical records:

- Segments from the same message can go in the same logical record.
- Segments from different messages cannot go in the same logical record but can share the same block.
- An input segment cannot span logical records (except SPAs).
- An output segment can span LGMSG logical records only if for a non-MFS terminal.
- Multisegment messages can use logical records from both the short and the long data sets.
- Segments from a single message are chained together across logical records but not within a logical record.

Ideally, the message queue pool is large enough to handle the peak message rate without any I/Os to the message queue data sets.

If the LRECL sizes of the SHMSG and LGMSG data sets are not in correct proportion to each other to handle the mix of message lengths, it means that relatively short messages are using logical records from the LGMSG queue. This in turn means that buffers holding these LGMSG queue records are significantly underfilled and likewise, the message queue pool itself, which holds these buffers, contains much wasted space.

Message queue I/Os that occur during transaction processing increase program elapsed time by the duration of the IWAIT. These I/Os are indicated on the IMS monitor region IWAIT report (Figure 25 on page 80), and the occurrences should be minimal.

Factors in the IMS system that impact message queue performance are multisegment messages, remote printer output, multiterminal output, program-to-program switches, terminal operator paging, zero-priority messages for BMPs, and conversational processing SPAs. The time on the queue for an input message is the duration from the arrival of the message in the system until the application program sync point.

Conversely, an output message is on the message queue from the ISRT call until the SNA acknowledgment (DR2) is received from the terminal. To dequeue a message, the message must be available in the queue pool. Single-segment messages are more efficient than multisegment messages for the queue manager, MFS, the communications analyzer, and the application program. The
time-in-system and size of input and output messages are important factors in the allocation, and subsequently the performance, of the message queue pool. The IMS log transaction analysis utility and IMSPARS are good tools for monitoring input/output queue times.

9.9.3 Number of Message Queue Data Sets

IMS can have multiple message queue data sets. Up to ten data sets can be allocated for each of the LGMSG and SHMSG queues. You can have different numbers of each, but make all of the SHMSG queue data sets the same size, and likewise for the LGMSG queue data sets. The objective of this facility is to allow parallel I/Os to the message queues.

Many high-performance IMS systems do little or no I/O to the queue data sets during normal processing. Even at system checkpoints, IMS writes out only those messages that have remained untouched on a queue since the previous checkpoint. The need for parallel I/Os can theoretically occur if an IMS system, processing a very high message rate, stalls for some reason. The input messages still keep coming and can fill the queue pool, so IMS starts spilling the older ones out to DASD. When the system resumes message processing, messages may continue arriving from the network. The older messages on DASD have to be retrieved for processing, which in turn means writing out some later messages to make room in the queue pool. This creates serious contention on the queue data sets.

Putting this in perspective, a system with a 5 MB QPOOL, receiving messages of 500 bytes at a rate of 100 per second, could take up to 100 seconds to fill the pool (depending on queue LRECL sizes and other factors). Therefore, most systems with plentiful storage require a good sized QPOOL and just one data set each for SHMSG and LGMSG.

Storage-constrained IMS systems experiencing message queue I/Os during normal processing can exploit the multiple queue data sets facility, and two or three data sets can be allocated to each of the short and long message queues. Do not consider the maximum number, ten, as a recommended value.

9.9.4 Message Queue Hash Search Algorithm

Messages are identified by their DCB relative record number (DDRN). When a message is requested for processing, the DDRN must be translated by the queue manager to a buffer address. This is done using a hash table. The DDRN is hashed to give an entry in the hash table, and all messages that hash to that entry are chained together in the buffer pool.

9.9.5 System Checkpoints and the Message Queues

The message queue manager is designed so that it writes to the SHMSG and LGMSG queue data sets only when buffers are not available in storage. The one exception is that, at simple checkpoint, IMS sends on to DASD any messages that have remained in the pool since the previous checkpoint. This means that in general, no messages need to be deleted at checkpoint time. On the other hand, emergency restart must always restart from, at least, the last checkpoint but one (unless the last checkpoint was a SNAPQ), to ensure that every message that needs to be recovered is either on the queue data sets or in later log records.
The number, average duration, and maximum duration of system checkpoints is reported in the IMS monitor region summary report.

A special case is a SNAPQ checkpoint, where the queues are written out to the system log. This is separately reported on the IMS monitor general reports, in the section entitled “General IWAIT Time Events.”

The topic is discussed more in 8.6.2, “Monitoring Checkpoints” on page 152.

9.9.6 Number of Message Queue Buffers

Increasing the number of buffers normally reduces the number of I/Os per transaction. The number of buffers can be specified at IMS initialization time in the execution parameter QBUF= of the IMS procedure or in the default IMS.PROCLIB member DFSPBxxx.

If very few I/Os to the message queue pool data sets occur (fewer than one-half per transaction or 15 per second), the message queue pool is not a bottleneck.

Too many buffers can cause page faults in a storage-constrained environment. In this case, it is usually better to reduce the number of buffers to reduce paging and accept a higher I/O rate per transaction. Monitor this higher I/O rate until the elapsed time per I/O begins to increase the response time to an unacceptable level. The guideline for an acceptable I/O is 20 to 25 ms, but this could increase to approximately 35 ms before an increase in response time is noticed.

The reason that a high I/O elapsed time is better than page faulting is that a page fault suspends the control task until it is resolved, and therefore the control task cannot dispatch any other work. The average time to resolve page faults depends on the paging subsystem at your installation.

9.9.7 Message Prefixes

Every message contains a message prefix. The message prefix must fit into one MSGQ LRECL in nearly all cases (with one exception described below). Message prefix segments are always placed at the front of the message, ahead of the user data segments. User segments are limited to 32 KB, the block size of the long message queue, or (if spanning is not allowed), the long MSGQ LRECL (minus prefix overhead), whichever is less.

Table 14 on page 180 documents the message prefix sizes (in bytes) for the various releases of IMS/ESA.
MSC can span the prefix across QPOOL buffers (as of IMS/ESA version 5) when sending a message to another IMS. MSC does not send all of the prefix segments with the message. The basic, system, system extension, and workload manager prefix segments are not sent across MSC links. These prefix segments are rebuilt on the receiving IMS when necessary. No prefix segments are sent to any terminals, including ISC nodes, although information is extracted from prefixes in some cases and included in the VTAM FMH. Prefix segments are not sent to application programs, although data is extracted and placed into the I/O PCB by the message GU call. All other prefix segments are sent.

It is recommended that the QPOOL buffers be large enough for MSC to send the largest message without spanning, and so eliminate the need for multiple sends of a message. MSC sends only as much data as necessary, not the entire buffer.

When using MSC to connect IMS systems of different releases, it is important to consider all message types (ISC, APPC, OTMA, and others) and the prefix sizes that accompany them. We recommend the MSGQ LRECL and block sizes be identical across all of the IMS MSC systems. If this is not possible, make the LRECL and block sizes at least large enough to hold the message prefixes and segments without spanning (or at least without additional spanning).

IMS increases a prefix segment's size if necessary, when the message is received on another IMS (through a MSC link) but does not decrease it and truncate the data. This is in case the message is sent back and the prefix data is needed. IMS adds prefix segments, if needed, when the message is received on another IMS but

<table>
<thead>
<tr>
<th>Prefix Section</th>
<th>IMS 3.1 Size</th>
<th>IMS 4.1 Size</th>
<th>IMS 5.1 Size</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>All messages</td>
</tr>
<tr>
<td>System segment</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>All messages</td>
</tr>
<tr>
<td>MFS prefix</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>All messages if MFS is generated</td>
</tr>
<tr>
<td>System extension</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>All messages</td>
</tr>
<tr>
<td>Ext prefix header</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>All messages</td>
</tr>
<tr>
<td>Security prefix</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>All messages if RACF used</td>
</tr>
<tr>
<td>Workload manager</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>All messages</td>
</tr>
<tr>
<td>MSC</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>All input and all remote output, if MSC is generated</td>
</tr>
<tr>
<td>MSC extension</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>All if MSC is generated</td>
</tr>
<tr>
<td>ISC (LU 6.1)</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>Used if ISC message</td>
</tr>
<tr>
<td>APPC (LU 6.2)</td>
<td>N/A</td>
<td>138-512</td>
<td>138-512</td>
<td>Used if APPC message</td>
</tr>
<tr>
<td>OTMA</td>
<td>22</td>
<td>138-4K</td>
<td>128-4K</td>
<td>Used if OTMA message</td>
</tr>
</tbody>
</table>
does not delete a prefix segment received from another IMS that it does not need (again, in case it is needed on another IMS).

In general, the prefix segment size is the size indicated for that release (see Table 14 on page 180) or the size from the IMS where the prefix segment type originated, whichever is larger.

One common problem occurs when an IMS system is migrated to a new release that has larger prefix sizes and new prefix segment types. Messages that contain these new and larger prefixes may not fit on the message queues of the earlier release IMS. This can cause problems when formatting the message and delivering it to its destination, especially with MFS, which treats each spanned segment piece as a separate segment when mapping the message.

### 9.9.8 Short and Long Message Queue Logical Record Sizes

Specification of sizes for the short and long message queue LRECLs allows optimal utilization of the message queue pool. Define the LGMSG LRECL as a multiple of the SHMSG LRECL, but try not to exceed 2 KB. However, the LGMSG LRECL must be as long as the longest output message segment that is processed by MFS, plus 52 bytes.

Sizes greater than 2 KB normally make inefficient use of buffer space for QBLKS and SHMSG LRECLs.

Make the LGMSG queue block size and the queue pool buffer size large enough to hold just one LGMSG LRECL. If two or more logical records are allocated per block (and buffer), and that block is selected for building a message, then the block has to be read from disk if it is not already in the buffer pool. When there is only one logical record per block, IMS simply creates the new message in the buffer, without first having to read the block from DASD.

Further, when the block becomes completely empty, rather than having it age down the LRU queue, it is marked as LRU and immediately becomes available for reuse. This just about guarantees that no WRITE I/O will be necessary for the next requestor of a QPOOL buffer.

Conversational SPAs are also passed through the message queues, so their size is critical to message queue performance. Refer to the discussion of scratch pad area (SPA) sizes in 9.9.9, “SPA Size” on page 182.

Analyze average message lengths to determine the most efficient choice of SHMSG and LGMSG logical record size. The average message lengths and counts for each transaction and terminal are maintained in the SMB and CNT control blocks. These can be obtained from a dump of the IMS control region or from the IMS log checkpoint records. For conversational programs, SPAs are passed through the message queue, so SPA sizes are included in the average message-length calculations.

Another source of average message length is the IMS statistical analysis utility line and terminal report. The average sizes sent (S) and received (R) are calculated from the log X’01’ and X’03’ records. Because these sizes are calculated by line, it is difficult to separate short and long message lengths.
IMSPARS has a message queue utilization report that can be very helpful (see Figure 118 on page 359). The report indicates the relative use of the short and long queues, and actually recommends the best LRECLs for each of the two queue data sets.

The report is in three sections. The first section is for messages (single or multisegment) whose length is less than or equal to the short message LRECL size, so that the complete message can fit in a short LRECL. The second section is for messages longer than the short LRECL size but shorter than or equal to the long LRECL, so that the complete message can fit in a long LRECL. The third section is for messages longer than the long LRECL.

### 9.9.9 SPA Size

Large SPAs can be one of the greatest single contributors to poor message queue performance and, ultimately, to poor system performance. When a conversational transaction is entered, IMS places the associated scratch pad in the message queue preceding the first segment of the message. Therefore, estimates of message queue LRECL size must include SPA sizes. A SPA is compressed (repeated character elimination) both when it is logged and on the message queue.

If the SPA and the first segment of the message exceed the LGMSG LRECL size, then multiple LGMSG LRECLs are required. Specify a larger LRECL size if the transaction with the large SPA has a high arrival rate. This choice depends on the relative number of transactions requiring larger SPAs. Avoid large SPAs, if possible, in application design. Instead, store data on a database and pass it from one application to the next.

In summary, the important tuning considerations for the message queue pool are these:

- Optimize LRECL sizes to make efficient use of pool space, and balance I/O activity between the SHMSG and LGMSG data sets.
- Provide enough buffers to minimize message queue I/Os.
- Allocate the size of the LGMSG LRECL, if large SPAs are defined, to include the size of the SPA plus the length of the input message.
- Use fixed-length SPAs.

### 9.10 Expedited Message Handler Buffer Pool

Every transaction using expedited message handling requires a buffer for the input and output message. This buffer is referred to as the Fast-Path terminal buffer or the EMH buffer (EMHB). It is acquired from the EMHB pool. This pool is managed by the DFSPOOL storage manager and expands or contracts based on demand.

If the control region execution parameter EMHL places a maximum size on the EMH buffer pool that is too small, error messages are returned to terminals that cannot have their messages processed.
9.10.1 Buffer Allocation

The size of an EMH buffer can be specified:

- As a control region execution parameter, EMHL=
- On a Fast-Path program (IFP) APPLCTN macro
- On a Fast-Path potential TRANSACT macro
- On the TERMINAL macro for a static terminal

At IMS initialization time, IMS sets the system default EMHB size to the EMHL execution parameter value. If none is specified, it uses the largest value of all sizes specified on TERMINAL, TRANSACT and APPLCTN macros. If none of these is specified, IMS uses 2 KB for the system default EMHB size.

The first time after logon that a terminal actually uses a Fast-Path program, the input message is placed in a message queue buffer and the Fast-Path input edit/routing exit (DBFHAGU0) is called. If it is a Fast-Path potential transaction, the exit chooses whether to use EMH. When EMH is chosen, or it is a Fast-Path exclusive transaction, then the exit chooses the Fast-Path program by setting a routing code.

At this time, a Fast Path buffer is allocated in the EMHB pool. The size is set as follows: If there is a TRANSACT value and it is larger than the system default, use the TRANSACT value; otherwise use the system default size.

Note

There is a TRANSACT value only if it is a Fast-Path potential transaction defined with TRANSACT.....FPATH=(size).

Each subsequent Fast-Path transaction, exclusive or potential, is put into this allocated EMH buffer, if it fits. If a Fast-Path potential transaction arrives that is too big for the current buffer and its TRANSACT macro specifies a message size larger than the current buffer, then the current buffer is released and a new larger buffer is allocated. Otherwise, a DFS0444I message is sent to the terminal saying the input message is too big.

If the Fast-Path exit (DBFHAGU0) decides to route a Fast-Path potential transaction to the message queue, then IMS moves it from the EMH buffer into a buffer in the message queue pool. From this, we can draw the following points:

- When the EMHL execution parameter is specified, TERMINAL and APPLCTN buffer sizes are ignored.
- A Fast-Path potential transaction can always be processed if the buffer size is specified on the TRANSACT macro.
- The buffer allocated to a terminal is always the maximum size ever needed since logon.
- If EMHL is specified, the size of a terminal buffer must be the EMHL size or larger. If every Fast-Path exclusive transaction size is less than the EMHL size, then no Fast-Path exclusive transactions are rejected with DFS0444I.
9.10.2 Recommendations

We recommend the following:

- If a few terminals enter some Fast Path Potential transactions that are significantly larger than average, specify the EMHL value for the typical transaction, and code the larger buffer size on the appropriate TRANSACT macros.

- Otherwise, use only EMHL and specify the largest buffer size ever needed.

9.10.3 Buffer Release

When the terminal logs off, the EMH buffer is released. In the case of APPC/IMS implicit processing, the EMH buffer is allocated at the beginning and released at the end of the transaction.

9.10.4 EMH Buffer Monitoring and Tuning

General information about monitoring and tuning the dynamic pools can be found in 9.2.1, “Dynamic Pool Manager (DFSPool)” on page 162.

9.11 Message Format Pool

In order to edit a message, MFS needs two control blocks: a MID/DIF combination (on input from a terminal) or a MOD/DOF combination (on output to a terminal). These blocks are brought into the message format buffer pool (MFBP) as required and are maintained in the pool on a least recently used (LRU) basis. When pool space is required, the space holding the LRU blocks is freed until the required block can be read in.

9.11.1 MFS Block Fetch

The MFS data set is a partitioned data set (PDS) where each block is a member. However, in retrieving a block, the normal PDS sequential directory search is not done each time. At initialization time, IMS allocates space for directory entries in the buffer pool prefix sufficient for 10% of all the blocks in the MFS data set. As each MFS block is first referenced, its address is retrieved and then stored for subsequent use in the resident directory. If and when the resident directory fills, an additional amount of storage is acquired to hold directory entries for another 10% of the blocks. This is repeated as often as necessary, until all of the used blocks have their directory entries in storage.

To ensure that large numbers of directory entries can be efficiently managed, a hash table is also built that has a number of anchor points equal to one-eighth of the number of MFS blocks. The directory entries include a synonym pointer that chains all directory entries off each hash table entry.
Additionally, the MFS service utility (DFSUTSA0) can be run with the INDEX function to create a list of the disk addresses of user-specified blocks. This list is then used at IMS initialization time to prime the resident directory for the most frequently referenced blocks (up to a limit of 2339). The utility creates a member in the MFS data set called $IMSDIR, which contains the address and size of each block. At IMS initialization time, this member is read into the MFBP prefix, and its use eliminates the need for I/Os to read directory blocks.

If online change is used, the $IMSDIR entry ensures an immediate refresh of the active format directory entries and bypasses the need for IMS to issue BLDL the next time each format is referenced.

To find out which MFS blocks are most frequently read into the pool, and so qualify for inclusion in the resident block directory, the IMSASAP total system IWAIT detail report can be used. The IMS monitor communications IWAIT report also has the necessary information but, because it is unsorted, employing it to list the most frequently used format blocks is very laborious.

### 9.11.2 Prefetch and Immediate Fetch

An MFS block pair often includes the names of the next expected block pair. IMS attempts to prefetch anticipated blocks. The prefetch routine checks to see if the anticipated block is in the pool but not currently in use (in other words, is on the free blocks queue). If it is there, IMS moves it to the bottom of the queue and marks it most recently used. If prefetch does not find the necessary block in the pool, nothing else happens at that time. When a message requires a block that is not in the pool, IMS does an immediate fetch of the block.
9.11.3 MFS Buffer Pool Queues

When a fetch request is issued, IMS searches two queues before attempting any I/O. The queues searched are:

- Free blocks queue = blocks already in the pool but no longer in use.
- Immediate fetch requests queue = blocks fetched or having been immediate-fetched.

(There used to be two other queues related to prefetch. These are no longer used.)

The actual elements on these queues are called fetch request elements (FRE).

9.11.4 Fetch Request Elements

Before any request to read a format block can be satisfied, a FRE must be assigned, from which the block will be chained. FREs are 48 bytes each, and although their number is defined at IMSGEN (BUFPOOLS FRE=n), the number can be overridden by the FRE= control region execution parameter. FREs normally reside in the MFBP prefix. However, if insufficient FREs are defined, IMS dynamically creates an FRE in the buffer pool itself. A dynamic FRE is destroyed when the corresponding block is overwritten.

Before creating a dynamic FRE, however, IMS tries to steal one from the free blocks queue. Stealing from the free blocks queue means wasting pool space because the associated block becomes unusable even though the pool is big enough to hold it. It must be reread if needed again. In other words, a large pool is of little use if not enough FREs are available to keep blocks in it when MFS has temporarily finished with them.

If the amount of FREE space shown in /DIS POOL MFP is high relative to the size of the pool, it is often an indication that not enough FREs are allocated.

To determine from the IMS Monitor reports whether enough FREs are specified, look at the BLOCKS WASHED FOR FRE row on the Message Format Pool report (Figure 32 on page 95). This value should be zero. If not, increase the number of FREs.

9.11.5 MFS Pool

The display of the MFS buffer pool statistics indicates both size and space. Size is the total size of the pool and includes the pool prefix, which contains the DCBs, the statistics, the FREs (except dynamic FREs), the I/O areas for PDS directory reads, and the resident block directory. Space is the size of the dynamic area used for the MFS blocks themselves and dynamic FREs, if any.
Figure 62. MFS Buffer Pool

The minimum pool size is 256 KB but, in reality, the pool needs to be much larger (Figure 62). IWAITs for MFS pool space show in the IMS monitor communications IWAIT report if the pool is too small. Do not allow this to happen.

The IMS control task can tolerate page faults in this pool. IMS uses a PAGELOAD SVC to reference the required page when not in storage, so that the IMS control task is not suspended.

9.11.6 MFS I/O Performance

The MFS compiler tends to place MFS block pairs (MID/DIF or MOD/DOF) close to each other on the disk. Consequently, the I/O time for the second block of a pair is usually very low. The use of IEBCOPY to move or compress an MFS library keeps related blocks together because it copies members in physical sequence, unlike IEHMOVE, which moves the members in alphabetical order, thus separating related members.

IMS allows multiple MFS data sets to be concatenated when they all have the same DCB attributes. In this case, parallel I/Os can be achieved. However, be aware that with concatenated MFS data sets,

- Adding a member to one data set does not delete it from a different data set.
- The $$IMSDIR members from all the data sets are merged to form a composite list of directory entries.
- The first fetch of any member not in a $$IMSDIR requires searching in turn through each concatenation’s PDS directory until its location is found.
9.11.7 MFS Performance Considerations

The sharing of formats between applications is an application design consideration. Having fewer formats increases the probability that the format is already in the pool when required. The pool size requirement may also be lower if the formats are shared.

If large messages are sent to the terminals, investigate the DFLDs of these messages to ensure that either the FILL=NULL or FILL=PT parameter is specified. If these parameters are not specified, unnecessary blank characters could be transmitted over the link, which increases the response time to the end user.

In summary, the basic format pool tuning considerations for an existing system are these:

- Make the number of FREs enough to use all of the allocated pool.
- Share formats among applications.
- Specify FILL=NULL or FILL=PT for all DFLDs in the format definition.

The number of MFS I/Os is ideally less than 0.5 per transaction or fewer than 15 per second with an I/O service time of 20 ms. If the pool size is increased, be sure to increase the number of FREs to avoid dynamic FRE allocation.

Refer to the “Performance Factors” section in the “MFS Application Design” chapter in IMS Application Programming:Design Guide (SC26-8016) for additional information on MFS performance.

9.11.8 Test MFS

When a terminal is in test MFS mode, the required MFS blocks are read into the CIOP and no use is made of the MFS buffer pool. No statistics are maintained of test MFS usage.

9.12 Scheduling Pools

Five pools are associated with scheduling:

- PSB pools (PSBP and DPSB)
- PSB work pool
- EPCB pool
- DMB pool
- DMB work pool

I/O activity to ACBLIB or pool-space failures during application program scheduling indicate the need to tune these pools. A pool-space failure in any of these pools indicates that a request for space cannot be satisfied from space not already allocated (for PSBs, PSB work areas, or DMBs needed by scheduled application programs). Thus, the pool that caused the pool space failure is too small to handle
the concurrent dependent region activity and needs to be increased. The rule for
the starting size of these pools is that each one be large enough to contain the
largest unit (PSB or PSBW) in the system, times the number of active dependent
regions, plus 20% for possible fragmentation. For example, if the largest PSB work
area in the system is 30,000 bytes and 50 active dependent regions are active, the
PSBW pool minimum size would be:

\[
1.2 \times (30,000 \times 50) = 1,800,000 = 1.8 \text{ MB}
\]

This still may not be enough because of fragmentation. Therefore, monitor these
pools using the IMS Monitor, a /DIS POOL ALL, or IMSPARS to determine whether
the pool size is adequate. If the monitoring of the pools shows that the high water
mark of the pool is well below the allocated storage size, the size of the pool can
be reduced to approximately 8 KB above the high water mark. The exceptions are
the PSB and DPSB pools which attempt to use all storage available.

Execute IMSPARS to display the resource usage reports between the peak
checkpoints to determine whether some of the pools are nearing their maximum
size.

ACBLIB I/O activity can occur to the PSB and DMB pools if they are not large
enough to hold all of the PSBs, nonresident intent lists, and DMBs referenced by
the system. Make the pools large enough to minimize the number of I/Os, yet at
the same time small enough to avoid a substantial increase of the constraints on
real storage. ACBLIB I/Os are undesirable for DMBs, except at IMS startup or
following /DBR and /STA database commands.

Try to limit ACBLIB I/O to less than one per scheduling. Each I/O increases the
application program schedule time not only for this application but for others as
well. IMS block mover is a serial resource; therefore, other dependent regions may
have to wait to schedule when the block mover is busy loading the application's
control blocks or when dynamic allocation is holding a block mover latch.

If the I/Os per transaction to ACBLIB total more than one, this indicates a possible
inadequate allocation of the PSB or DMB pool (the PSBW, EPCB or DMBW pools
never experience I/O activity). The potential exists for a minimum of three ACBLIB
I/Os per scheduling: one I/O for the intent list (INT=), one for the PSB (PSB=), and
one for the DMB (DMB=). The number of I/Os for each PSB or DMB depends on
the size of each and the block size of ACBLIB. It is recommended that the ACBLIB
block size be 32 KB for a 3390 or 3380 device.

Unique considerations for each pool determine how much larger than minimum size
it needs to be. Remember that we recommend fixing the PSB pool size. Using the
/DIS POOL output as well as the IMS monitor reports, evaluate the size of each
pool as described in the following subsection.

### 9.13 PSB Pool

A PSB can be considered as three separate control blocks:

- The PSB prefix and PCBs
- The intent list
- The work area
The first two control blocks use the PSB pools and are stored in the ACBLIB, a PDS in which each PSB is a member (Figure 63 on page 190). The work area is dynamically created in the PSB work pool at scheduling time.

The intent list exists as a separate block within the PSB and is pointed to by a pointer in the user data portion of the PDS directory entry for that member.

9.13.1 PSB Pools

The PSB pool is one pool if the LSO=Y option is specified in the IMS JCL procedure. The PSB pool becomes two pools if the LSO=S option is specified: one in ECSA, which is still named the PSBP pool, and the other in the DL/I separate address space, which is named the DPSB pool.

The sizes of these two pools can be specified in the IMS system definition, in the IMS.PROCLIB member DFSPBxxx, or on the IMS control region JCL procedure as CSAPSB= and DLIPSB=. They may still be specified even if the LSO=S option is not chosen, but then the specification of these pools is ignored. Conversely, if the PSB pool definition is used and LSO=S is specified without the CSAPSB and DLIPSB specifications, the pools are allocated as 20% of the size specified in the PSB= parameter in CSA, and 80% of the size specified in the DL/I separate address space. Allocate the pools using the CSAPSB and DLIPSB parameters. Do not allow them default to 20% and 80%.

If LSO=S is specified, the PSBP pool in CSA contains the IOPCBs and alternate output PCBs for all transactions. If Fast Path is defined in the system, the Fast Path PCBs also reside in this pool. The DPSB pool in the DL/I separate address space contains the full-function DL/I PCBs for all transactions.
If the LSO=S option is not specified, the PSB pool, which resides in ECSA, contains all the PCBs referenced by the transactions being executed.

9.13.2 Resident PSBs and Intent Lists

An intent list requires an I/O when read from the ACBLIB (Figure 64), but all the intent lists are made resident in an area of storage with a CDE name of DFSINTRS if the default control region parameter RES=YES is specified (Figure 65).

An actual PSB requires at least one I/O when read from ACBLIB. However, if the RES=YES control region parameter is specified, PSBs with the resident option (RES) specified in their APPLCTN macros are loaded at IMS initialization time into a separate area and are available without I/O at program scheduling time. This area is called DFSPSBRS.

---

![Diagram of PSB pool structure with RES=NO Option](image1)

*Figure 64. PSB Pools with RES=NO Option*

![Diagram of PSB pool structure with RES=YES Option](image2)

*Figure 65. PSB Pools with RES=YES Option*
The resident option can be used to keep high-volume PSBs in storage. Additionally, you may consider using the resident option for WFI transactions because the WFI PSB remains active for the duration of the WFI application. Storing high-volume PSBs in the PSB pool may contribute to space fragmentation.

Resident PSBs can also be appropriate for preloaded programs if the reason for preload is simply to gain the best possible performance.

The option is also useful when a PSB is much larger than the most of the other PSBs. It avoids the premature deleting of inactive PSBs, as explained in 9.13.4, “PSB Pool Space Management.”

If online change is used, any changes to a resident PSBs cause the PSB to cease being resident. It uses the PSB pool until the next IMS restart.

Resident PSBs cannot be page fixed in the IMS DFSFIXxx member using the POOLS parameter, but they can be included in the module name fix list as DFSPSBRS.

Similarly, resident intent lists can be fixed by specifying DFSINTRS in the module name fix list.

### 9.13.3 PSB Scheduling

IMS always tries to avoid ACBLIB I/O. Thus, when a serial PSB is needed (only one copy is allowed at a time):

- IMS first checks to see if it is a resident PSB; if so, the resident copy is used.
- IMS then checks whether if the required PSB is already in the pools, brought in by a previous scheduling and no longer in use. If so, it is reused.
- Otherwise, the PSB is read from the ACBLIB.

In the case of parallel scheduling, IMS first looks for the necessary PSB in storage:

- If the PSB is resident and not in use, the resident copy is used.
- If the PSB is resident and already in use, an additional copy is made in the PSB pools by cloning the resident copy.
- If there is no resident copy, but at least one copy already exists in the pools, then an additional copy is made by cloning an existing copy.
- If no copy exists in storage, the PSB is read from the ACBLIB.

### 9.13.4 PSB Pool Space Management

When a PSB has to be cloned or read in, space must first be found in the pools. Free space in the PSB pool is managed by control blocks called *free-area or free/allocated queue elements, FAQEs.* These are chained together in order of decreasing size of the free space that they control. If the largest space is insufficient for the new PSB, then no scanning of the free area FAQEs is necessary. Otherwise, the most appropriate chunk of space will be found.

In most cases, sufficient free space is not available, so space must be made by buffer stealing. Buffer stealing starts by examining the least recently used inactive

---

11 FAQEs are used to manage both free space and allocated space.
PSB in the pool. Then, calculates how much space would be created if that PSB were deleted and includes any free space that exists immediately before and after the PSB. If this does not provide enough free space, then up to two passes are made of the inactive PSBs on the LRU chain, looking for:

1. An inactive PSB that is not the only copy in the pools and which, if deleted, would (with any contiguous free space) provide enough room for the new PSB.
2. An inactive and only copy of a PSB in the pools which, if deleted, would (with any contiguous free space) provide enough room for the new PSB.

If this fails to find space, more drastic action is taken. Each PSB on the LRU chain (starting at the oldest, of course), is freed in turn until enough contiguous free space is created. This can result in the destruction of a great deal of the allocated but inactive PSBs within the pools.

Consequently, the scheduling of an unusually large PSB that is not frequently used can cause a number of other inactive PSBs to be prematurely deleted from the pool.

### 9.13.5 Dynamic PSB Option

The dynamic PSB option (DOPT parameter on the APPLCNT macro) should not be used on production IMS systems. With this option, a BLDL is performed and a new copy of the PSB and the intent list are loaded from the ACBLIB each time it is scheduled. This can take a considerable amount of time if the ACBLIBs being searched have a large number of members. In a production system, this could have a significant impact on the scheduling of other transactions, especially considering the long time that the block loader latch is held.

A DOPT PSB is deleted from the PSB pool as soon as the program terminates.

### 9.13.6 PSB Pool Monitoring

PSB pool statistics are written to the log and reported by IMSPARS. The required report is the internal resource usage—variable pool statistics report. An example is shown in Figure 66 on page 195.

### 9.13.7 PSB Pool Tuning

The major performance-tuning considerations for the PSB pool are:

- Make intent lists resident.
- Define a PSB pool large enough to contain all of the active PSBs that are needed for each concurrently scheduled dependent region.
- Evaluate the use of resident PSBs for high volume or WFI transactions and long-running prime-shift BMPs.
- Page fix the PSB pools.

---

12 For normal PSBs, the BLDLs are all issued at IMS initialization time.
9.14 Extended PCB Pool

When any PSB containing Fast-Path PCBs is scheduled, extended PCBs are built in the EPCB pool.\(^{13}\)

The formula for calculating the size of the EPCB pool can be found in the *IMS/ESA V5 Installation Guide: Volume 2 (SC26-8024)*, in the section on the BUFPOOLS macro:

\[
EPCB = (\text{number of TP-PCBs} \times 28) + \\
(\text{number of MSDBs} \times 76) + \\
(\text{number of DEDB SENSEGs} \times 124) + 132
\]

This typically amounts to 2 KB or less per MPP region. The actual EPCB space requirements for each PSB are listed by the ACBGEN utility.

If the EPCB pool is too small, MPP scheduling failures (internal conflicts) occur and are reported on the IMS monitor Reports report (see Figure 71 on page 218).

IMSPARS reports on the actual amounts used in its internal resource usage—variable pool statistics report. An example is shown in Figure 66 on page 195.

9.15 PSB Work Pool

The PSB work area required by each active PSB is allocated from this pool when the PSB is scheduled. I/O never occurs for this pool. Using the output from a /DIS POOL PSBW command issued just before IMS terminates, reduce the size of the pool to the maximum (High) amount used. Monitor this new pool size for a few days to substantiate that it is the correct size and that PSTs are not waiting for space in this pool.

Another way to calculate the size of the PSBW pool is to scan the ACBGEN output, which shows the largest PSB work area. If the largest PSB work area found on the ACBGEN is for an active PSB, this size can be used for the calculation of the size of the PSBW pool.

Calculate the size of the PSBW pool by multiplying the size of the largest PSB work area times the number of dependent regions, plus 20%. This is the maximum size of the pool. Monitor its use daily before shutdown or every 24 hours to determine whether the size needs to be changed. If PSBW pool space failures occur (see Figure 71 on page 218), increase the pool size.

IMSPARS reports on the actual storage used in its internal resource usage—variable pool statistics report. An example is shown in Figure 66 on page 195.

The major performance tuning consideration for the PSBW pool is to make it large enough.

---

\(^{13}\) Some IMS manuals claim that EPCs are required only by MPPs, but this is no longer the case.
IMS INTERNAL RESOURCE USAGE

INTERVAL : 23.38 (HHHH.MM.SS)

<table>
<thead>
<tr>
<th>Pool Statistics</th>
<th>Count /Transact /Second</th>
</tr>
</thead>
</table>

**DMB POOL STATISTICS (DLDP)**
- Bytes in DMB Pool: 122,880
- Bytes Allocated at End of Report: 39,680
- Maximum Bytes Ever Used: 39,680

**PSB POOL STATISTICS (DLMP)**
- Bytes in PSB Pool: 921,600
- Bytes Allocated at End of Report: 397,560
- Maximum Bytes Ever Used: 413,560

**PSB POOL (DLI/SAS LOCAL) STATISTICS (DPSB)**
- Bytes in PSB Pool: 1,699,840
- Bytes Allocated at End of Report: 1,632,528
- Maximum Bytes Ever Used: 1,688,544

**PSB WORK POOL STATISTICS (PSBW)**
- Number of Bytes in Pool: 3,670,016
- Number of Bytes Allocated at End of Report: 51,608
- Maximum Bytes Ever Used: 203,952

**DMB WORK POOL STATISTICS (DBWP)**
- Number of Bytes in Pool: 32,768
- Number of Bytes Allocated at End of Report: 0
- Maximum Bytes Ever Used: 7,072

**EPCB POOL STATISTICS (EPCB)**
- Bytes in EPCB Pool: 491,520
- Bytes Allocated at End of Report: 3,976
- Maximum Bytes Ever Used: 5,272

Figure 66. IMSPARS IRU Variable Pool Statistics
Make the DMB pool large enough to contain all IMS database DMBs. If the IMS monitor is activated after all databases are opened, any IWAITs for DMB= indicate that the DMB pool size is too small or a /DBR was issued earlier for the database and it is now being referenced again. Each time a DMB has to be loaded into the pool, and adequate free space is not available in the pool, one or more databases not currently being referenced must be closed and their DMBs deleted to make pool space for the new DMBs being referenced. After the new DMB is loaded into the pool, the database must be opened. Opening and closing of databases suspends either the control region or the DL/I separate address space, depending on the specification of the LSO= option, and increases DL/I call elapsed time. Page faulting in the DMB pool is preferable to database open/close.

Make the DMBs for all databases resident. Resident DMBs are loaded into a pool called **DFSDMBRS**. If all the DMBs are made resident, the inactive DMBs can be allowed to page out without significantly affecting the active DMBs, but paging them back into storage can suspend processing in the control region or the DL/I separate address space. Resident DMBs can be page fixed by fixing module name DFSDMBRS.

If all DMBs are made resident in the IMS system definition, specify the size of the DMB pool as 12 KB. This allows a DMB not specified as resident in the IMS system definition to be loaded into this pool. Monitor this pool consistently to determine whether nonresident DMBs exist.

If online change is used for the system, the DMB pool must be large enough to contain all of the resident DMBs that are to be changed by online change. The resident DMBs that are changed by online change are closed, and the new DMB is loaded into the DMB pool when referenced again. It is not reloaded into the resident DMB pool.

Using the output from /DIS POOL, check whether all of the DMB pool is being used. Be sure the full IMS system has been running long enough that all the databases are open. If there is more than 8 KB of free space in the DMB pool, reduce the free space to 8 KB and monitor the size of the pool again.

**IMSPARS** also reports on the actual storage used in its internal resource usage—variable pool statistics report. An example is shown in Figure 66 on page 195.

The major tuning considerations for the DMB pool are these:

- Make all of the DMBs resident.
- Allow enough space in the DMB pool for online changed DMBs.
- Do not underallocate.
9.17 Database Work Pool

This pool called **DMBW** is used as a work area for the DL/I call analyzer, delete, and retrieve modules. Pseudo-abends occur if too little space is available in the pool. The size 2 KB per PST is usually adequate for most systems. The largest space use observed has been 24 KB, but the majority of installations seem to use less than 8 KB.

IMSPARS reports on the storage amounts used in its internal resource usage—variable pool statistics report. An example is shown in Figure 66 on page 195.

The major performance tuning consideration for the DMBW pool is to avoid underallocating.

9.18 Exclusive Control Enqueue/Dequeue Pool

This pool is used to store control information used by the IMS internal lock manager.

9.18.1 Overview of Program Isolation ENQ/DEQ

The exclusive control (PI) enqueue pool is used to hold the enqueue elements created for program isolation and internal IMS enqueue requests. Its purpose is to control usage of resources that can be employed by more than one user at a time. The control routines are called **enqueue/dequeue**. In IMS, the enqueue/dequeue function is supported by routines called **exclusive control**, part of the program isolation facility. The exclusive control function is called by various DL/I modules: open/close, the OSAM buffer handler, the VSAM interface, and HD space management.

9.18.2 Control Block Structure

Figure 67 on page 198 shows the three types of 48-byte control blocks used, which are built in the exclusive control ENQ/DEQ pool:

- **QCB**: Represents a resource (RBA in a data set, buffer, for example)
- **QEL**: Represents a requestor (PST)
- **REQBLK**: Represents a request for a resource by a requestor

The example shows two users, PST1 and PST2. PST1 has two segments enqueued, represented by QCBx and QCBy, chained by the request blocks REQBLK1A and REQBLK1B. The diagram does not indicate the levels of enqueue, but a possible explanation is that PST1 holds QCBy, but is waiting for QCBx, which is held by PST2 through REQBLK2.

When program isolation decides that an RBA needs to be enqueued, it hashes the DMB number, DCB number, and RBA to produce a value in the range 0 to 255, which is then used to identify the hash table entry to address a QCB. If the QCB does not exist, one is taken from the chain of free QCBs. If no free QCBs are available, IMS issues a GETMAIN for storage according to the increment value specified in the execution parameter PIINCR and creates a chain of free QCBs. If the QCB addressed already exists, it either represents the same segment or a
synonym. In the latter case, a new QCB is created and the synonyms are chained together in descending sequence of RBA/DMB/DCB.

When a program releases a segment (for example, GN root processing releases the previous root when it enqueues on the new root), it dequeues the segment using the hash table. However, changed data remains enqueued until sync point, when all the QCBs associated with a QEL are freed by purge. The maximum size of the pool depends on the total number of enqueue requests outstanding at any one time by all dependent regions.

The number of enqueue elements in the pool for a region depends on the type of database activity. That is, a DLET or ISRT call typically requires more elements than a REPL or GN call. More important, however, is the length of time the enqueue request is outstanding. The IMS system is designed to be transaction oriented. Batch message or message processing programs that execute hundreds of DL/I calls between synchronization points have many more enqueue requests and hold them much longer. Online systems with transaction profiles of less than 30 DL/I calls and no BMPs normally require less than 28 KB of pool space. If BMPs that require large amounts of storage are processing and virtual storage is critical, then more frequent checkpoint calls must be issued by the BMPs. The duration of the enqueue can also have a severe impact on performance if other programs are waiting for the enqueued resource.
A PSB containing a PCB with the exclusive processing option (PROCOPT=E) or get only option (PROCOPT=GOx) does not take locks when processing with that PCB (see 12.5, “PROCOPT=GOT with DBRC SHARECTL” on page 295).

9.18.3 Monitoring PI Enq/Deq Pool Usage

The IMSPARS resource usage report heading “Program Isolation Enqueue/Dequeue Statistics” lists the maximum bytes used and statistics about the activity against the pool.

The IMSASAP enqueue/dequeue trace report provides more detailed information. For each ENQ that results in the requestor queueing for the resource rather than being given it, the requestor IWAITs until the resource is freed. This report gives information on each such IWAIT and its duration. If only one requestor is waiting, then this report has one line that indicates the time the IWAIT started, the database and segment code of the segment involved, the duration of the IWAIT, and the active programs. The waiting program is marked with an asterisk.

If, while a resource is held, two or more requestors have to IWAIT on it, then this report will have two lines for each enqueue. The first line indicates the time the IWAIT began, the segment type involved, and the program that is IWAITing. The second line, which may not immediately follow the first, contains END in the segment code and shows the time the IWAIT finished and its duration.

Similar trace information can be obtained from the log or external trace data set if the PI trace has been started (whether by an option in the DFSVSMxx member or by a command). To list all locks, use the file select and formatting print utility (DFSERA10) with the DFSERA40 exit. To list only the contentions, use the PI trace report utility (DFSPIRP0).

There is no way to page fix the enqueue pool. The pool resides in the extended private area of the control region or the DL/I separate address space. If Fast Path is included in the system, the enqueue storage is allocated in ECSA. The pool can increase in size up to the maximum specified by the execution parameter PIMAX, but it is never decreased. Define the maximum size of the pool large enough so that U0775 abends do not occur. If these abends do occur, then increase the maximum size of the ENQ storage. Review the use of the LOCKMAX parameter as well (see 12.2.5, “Program Isolation” on page 284).

If the IRLM is used for PI control, then no use is made of the PI ENQ/DEQ pool. The ENQ/DEQ control blocks reside in the IRLM address space.

9.19 IRLM

The IMS resource lock manager (IRLM) is an external lock manager that can also be used as a global lock manager in a data-sharing environment.

Unless you are planning to migrate to a data-sharing environment, it is suggested that you use the previously discussed IMS internal lock manager, PI ENQ/DEQ.
9.19.1 IRLM Dispatching Priority

Give IRLM a very high dispatching priority, higher than that of any other IMS address space and probably higher than even VTAM. The IRLM TCBs are used very little, but when they are, it is essential that they have top priority.

9.19.2 Execution Parameters

In the IRLM procedure DXRJPROC, the key parameters for performance are:

- **PC=NO**

  With this option, locks are built in ECSA and accessed directly using cross-memory services. Using the alternative (PC=YES) requires more CPU to access the locks, which are kept in private IRLM storage.

- **MAXCSA=n**

  This is the maximum size of ECSA storage needed to hold the IRLM locks (in megabytes) and is mandatory with PC=NO. Take the value used with PI (PIMAX parameter) and multiply it by at least six to get MAXCSA.

- **DEADLOK=(m,n)**

  IRLM checks periodically to see if any wait is due to a deadlock. The first DEADLOK parameter indicates the minimum number of seconds that must elapse before a lock wait is investigated by the IRLM to see if it is a deadlock. The second parameter indicates how often to check for global deadlocks in a data-sharing environment.\(^\text{14}\) The value is specified as a multiplier of the first parameter.

  The default values are much too long. It is recommended that you use:

  \[
  \text{DEADLOK}=(1,1) \text{ or } (2,1)
  \]

9.19.3 IRLM Monitoring

IMS keeps summary IRLM statistics in the X’4521’ log records, which can be printed with the IMS file select and print formatting utility (DFSERA10).

Detailed trace information can be collected and printed exactly as for PI ENQ/DEQ, as explained in 9.18.3

9.20 Log Buffer Pool

The general characteristics of the IMS logging subsystem are discussed in 2.7, “IMS Logging” on page 31. The section considers how performance related to log buffers is monitored and tuned.

\(^\text{14}\) Note that this is not used to detect deadlocks between DB2 and IMS. In this case, DB2 experiences a timeout rather than a deadlock.
9.20.1 Monitoring Logging Performance

IMS keeps statistics about logging, which are best reported by IMSPARS. An example can be seen in Figure 114 on page 355. The report shows 3.48 check writes per transaction. This is high because this system uses MSC, which incurs additional forced writes to the log. In fact, many of these forced writes are combined and the resulting number of WADS I/Os per transaction is 1.31. On average, each transaction writes out 4.17 segments of 2 KB each to the WADS, which amounts to a little over 8 KB of log data per transaction. This is about one-third of a log buffer per transaction and, as expected, the OLDS I/O rate is 0.37 per transaction.

The OLDS logging rate is also shown as 4.25 I/Os per second. Assuming a 26 KB log buffer size, this amounts to 111 KB/second. To put this in perspective, some customers measure at over 3 MB/second.

The most relevant numbers in terms of tuning are in the “Logical Logger” section on the rows headed “Buffer Waits.” This part of the report shows the number of times that all the buffers were filled and waiting to be written when a component of IMS was waiting to put a new log record in a buffer. The first value (CHKPT INVOKERS) is effectively the number of IWAITs experienced by the system checkpoint process (these are not seen as IWAITs by the IMS monitor). This may well be other than zero even when no problems are perceived at checkpoint. However, the second number (NON-CHKPT INVOKERS) represents buffer waits when checkpointing is not in process, and is expected to be zero. If the value is nonzero, increase the number of log buffers.

9.20.2 Reducing the Volume of Log Data

Reducing the amount of log data produced is generally beneficial:

- Less resource is used in collecting the log data.
- Less contention for log buffers is experienced.
- OLDS switches occur less often, hence DBRC activity is reduced.
- Less data is processed by change accumulate or database recovery utilities.

There are several ways of achieving a reduction in log volumes:

- Avoid using the OLDS for writing trace data; use the external trace facility instead.
- Ensure that the Fast Path LGNR parameter is effective (see 9.20.4, “Logging Considerations When Using Fast Path” on page 202).
- Reduce system checkpoint logging volumes (see 8.6, “IMS System Checkpoints” on page 151).
- Use nonrecoverable databases.

9.20.2.1 Nonrecoverable Databases

IMS provides a way of reducing the amount of log activity by allowing a database to be registered with DBRC as nonrecoverable. If this option is specified, IMS does not log any after-images of the database changes that take place. However, before-images of any changes are logged, as these are required for dynamic backout. The before-images are kept only on the OLDS and are not archived to a SLDS.
Typical uses of a nonrecoverable database would be:

- Work database for CSP
- Scratch pad area database holding noncritical information that is passed between application programs

A database is registered with DBRC as nonrecoverable by using the NONRECOV option on the INIT.DB or CHANGE.DB commands. Using nonrecoverable databases for heavily updated work or scratch pad databases can significantly reduce log volumes. When used as a scratch pad area instead of using the SPA itself, a nonrecoverable database can also be very beneficial in reducing the load on the message queues.

### 9.20.3 Log Buffer Tuning

The objective here is to avoid the situation where all the buffers fill up faster than they can be written out, which is most likely at system checkpoint time. This objective might be achieved by:

1. Increasing the number of log buffers, perhaps up to 50.
2. Writing the log data faster to the I/O subsystem.
   - Use faster DASD for OLDS.
   - Use DFW for OLDS.
   - Use a DASD control unit with more nonvolatile storage.

DFW can work only when the data being written fits in the NVS. Consider the case where an IMS checkpoint is taking 5 seconds. It is possible that most of this time is spent doing real writes to DASD if there is a shortage of NVS. At a data rate of, say, 4 MB/second, this represents 20 MB of data being written. If there were an extra 20 MB of NVS, the data could be written directly to cache and NVS at channel speeds in about 2 seconds.

### 9.20.4 Logging Considerations When Using Fast Path

Fast Path builds all its log records at application sync point time. To make this possible for DEDB updates, IMS has to record, at DL/I call time, the location in the buffer where the change took place, and the length of the changed data. This information is stored in the buffer header, and a number of 4-byte slots are reserved for this purpose. The number of slots is defined by the control region execution parameter, LGNR=.

Each DEDB update log record has a log header of 104 bytes. So if two database changes occur within 104 bytes of each other, less log data is produced by logging both changes (plus the intervening unchanged data) in a single log record. Each time a DEDB buffer is updated, IMS checks whether the new change is within 104 bytes of a previous one. If so, IMS simply adjusts the contents of an already-used slot in the buffer header.

If the number of slots (LGNR) proves to be insufficient to record all updates more than 104 bytes apart, IMS consolidates of the information in the slots by setting each slot to control updates within 208 bytes of each other. If this still proves insufficient, consolidation is done one final time using the value 416 bytes. Finally, if the number of updates more than 416 bytes apart is greater than the LGNR value, then IMS logs the entire CI.
Consolidation, if done frequently, unnecessarily requires CPU resource. Logging the entire control interval, if done regularly, could substantially add to the volume of log data. Therefore, both are to be avoided.

With small Fast Path buffer sizes (most customers use 4 KB or 8 KB), the guaranteed safe option is to set

$$\text{LGNR} = \frac{\text{Fast Path buffer size}}{104}$$

If you use large buffer sizes (12 KB or more), consider using the maximum value, LGNR=99, unless the amount of updating is very limited (logging of entire 12 KB or more CIs is not desirable!)

Following this guideline, a Fast-Path user with 2000 buffers of 4 KB would specify LGNR=40. The required amount of page fixed storage is

$$2000 \times 40 \times 4 = 640,000 \text{ bytes}.$$  

Note

Do not allow the LGNR parameter to default. The default value of 7 is much too small.

### 9.20.5 Monitoring Fast Path CI Logging

The Fast Path log analysis utility (DBFULTA0) reports on the effectiveness of the LGNR parameter. The utility is described in the IMS/ESA V5 Utilities Reference :System (SC26-8035) manual. The relevant LGNR evaluation and PSBs responsible for consolidations and full control interval writes are detailed in the reports entitled “Overall Summary of Resource Usage and Contentions for All Transaction Codes AND PSBs” and “Recapitulation of the Analysis Report” of this utility.

### 9.21 IMS Monitor Buffer Pool

The IMS monitor collects IMS internal data and writes it out to the monitor data set; consequently, it requires a buffer pool. If block size is not specified on the IMSMON DD statement or the DFSMDA dynamic allocation statement, then IMS uses the minimum block size for the IMS monitor data set. For DBCTL, the minimum is 1048 bytes. For IMS TM, the minimum is the larger of the long message logical record length plus 24 bytes, or 1048 bytes. In all cases, the block size is rounded up to a multiple of 8 bytes. If BUFNO is not specified on the IMSMON DD statement or the DFSMDA dynamic allocation statement, IMS uses only two buffers.

These default values for block size and number of buffers generally provide inadequate buffering. This often causes IMS to wait when attempting to write a monitor record. In other words, switching on the IMS monitor can degrade performance, with the consequence that the reported performance is not in fact the true performance of the system. Larger block sizes and more buffers eliminate this bottleneck.

The following are recommended:
Always specify the block size parameter. For DASD, use half-track blocking. This is BLKSIZE=27992 for 3390s. For tape, use a 32 KB block size. This is BLKSIZE=32768.

Always specify the BUFNO parameter. Make the value at least 5. For systems doing over 100 transactions per second or with more than 10 concurrently executing BMPs, specify a larger value. BUFNO=20 is adequate for any system.

For example, you could use the following DD statement:

```plaintext
//IMSMON DD DSN=IMS.IMSMON,DCB=(BLKSIZE=27992,BUFNO=10),DISP=SHR
```

IMS monitor buffers reside in ECSA. The use of ten buffers with a 27992 block size requires less than 280 KB of ECSA.

### 9.22 Automated Operator Interface Pool

The IMS automated operator exit - Type 2 (DFSAOE00) can use callable services to send messages to automated operator applications (typically BMPs), which use GMSG DL/I calls to retrieve them. These messages do not get queued on the IMS message queues, but instead are kept in the automated operator interface pool (AOIP).

The Type 2 AOI is used by some non-IBM automated operator products and is essential in the CICS/DBCTL environment, where there are no message queues.

The AOIP is a dynamic pool managed by DFSPOOL, which expands and contracts as required. Information about monitoring and tuning the DFSPOOLs is given in 9.2.1, “Dynamic Pool Manager (DFSPOOL)” on page 162.

### 9.23 Work Area or General Pool

The pool WKAP is used to provide work area for the /ERE command and simple checkpoints, and for general IMS working storage.

The only way to see how much storage has been used in this pool is by issuing a /DIS POOL command. The specific command for the pool WKAP is:

```plaintext
/DIS POOL MAIN
```

![Figure 68. /DIS POOL MAIN](image)

The total size (Figure 68) is indicated by “Size” and the high water mark by “High.”

The IMS monitor reports pool shortages in the region IWAIT report, indicated by IWAITS for STG. POOL=TPPL.

Its size is specified only in the control region execution parameter, WKAP. Never specify it as less than 20 KB, even though it may not seem likely to ever need this
much. The demands by /ERE are unpredictable. The default value, 5 KB, is much too small.

### 9.24 Fast Path Work Pool

The Fast Path work pool (FPWP) is managed by the DFSPOOL storage manager and dynamically expands and contracts according to demand. It was originally created to support the first implementation of DEDB high speed sequential processing (HSSP), and in particular for the HSSP image-copy option. At that time, IMS logged information about which image copied CIs had actually been updated by the HSSP application (because the image copy was in effect, a log of those updates). A bit map was used to record this information and was logged in X'5947' log records. The FPWP was used for building the bit map.

IMS/ESA Version 5 has a completely new HSSP image copy implementation called **HSSP asynchronous image copy**. All updates are logged with normal log records, and the X'5947' log record, together with the bit map it contained, is no longer used. However, the FPWP has come to be used by any of the Fast Path modules whenever they require work space. It is mostly, but not exclusively, used at IMS restart time.

General information about monitoring and tuning the dynamic pools can be found in 9.2.1, “Dynamic Pool Manager (DFSPOOL)” on page 162.

### 9.25 Communications External Subsystem Pool

The CESS pool is a dynamic pool managed by DFSPOOL, which expands and contracts as required. Information about monitoring and tuning the DFSPOOL is given in 9.2.1, “Dynamic Pool Manager (DFSPOOL)” on page 162.
Chapter 10. Scheduling and Dependent Region Considerations

The basic approach to controlling an online IMS system with loaded queues is to group application programs into a reasonable number of so-called dependent regions which can execute concurrently. The performance of the dependent region will affect the entire IMS system greatly. We discuss some performance topics in the following sections.

10.1 Dependent Region Problem Identification Chart

Figure 69 shows ways of dealing with dependent region scheduling problems.

**10.2 Program Scheduling**

Scheduling is the process of selecting a transaction from the input queue and associating it with the appropriate program to be loaded in an available dependent region. The IMS scheduler is driven in either of two situations:

- A new transaction arrives in the system and at least one dependent region is available.
- A dependent region finishes processing its previous transactions and becomes available for other work.
The IMS scheduler can execute either under a control region task (serialized within the control region) or under any of the message processing region tasks (parallel scheduling).

Transaction selection is based on priority within class, where:

- A transaction has a class and a priority within that class.
- A dependent region has up to four classes associated with it. These classes are specified as dependent region execution parameters; the sequence defines the relative priority of the classes for that region only.

### 10.2.1 Class Scheduling

The feature of IMS that allows transactions to execute in different regions is known as **class scheduling**. This feature is implemented in most installations. It allows low-priority transactions and long-running transactions to be isolated from short-running and high-priority transactions.

Maximum transaction throughput and a balanced region workload can be achieved with no transaction class or priority scheduling. However, class scheduling is necessary when a particular class of transactions requires a better response time at the expense of lower-priority transactions in the system.

Carry out the class scheduling of high priority transactions so that multiple high-priority regions can execute the transaction. Do this unless there is a special reason, such as program isolation conflicts (PI enqueues) that negate this option. Using multiple regions for the high-priority work reduces the chance that the transaction waits on the input queue. Multiserver systems generally produce shorter waiting times than single-server systems.

The option is available to assign lower-priority transaction classes to a high-priority region in case no higher-priority transactions are available to process. The fallacy in taking this option is that, if a long-running, lower priority-transaction is scheduled in the region, and a higher-priority transaction arrives in the queue, the higher-priority transaction may have to wait on the message queue until the lower-priority transaction finishes executing. This is a risk that you need to assess for your installation.

IMS tends to use all dependent regions over time if they support active transaction classes. IMS arranges the PST queue by placing the PST that just completed at the end of the queue. IMS then tries to schedule any transaction in the first available region that has the class of the transaction being scheduled. This technique uses a round-robin type of region usage.

For very high-volume transactions, consider EMH (Fast Path) first. The major requirements are that the transactions be in response mode and that both the input and output messages be single segment. EMH supports both serial and parallel scheduling of transactions.

Alternatively, dedicate one or more full-function message regions to the transaction, and define it as wait-for-input (WFI) to minimize the scheduling costs. WFI is discussed on page 209.
10.2.2 Scheduling Classes and the MVS Work Load Manager

As explained in 3.2, “The MVS Workload Manager” on page 39, for the WLM to be effective in allocating MVS resources to the IMS system, all the transactions processed by a dependent region must be of similar profile and importance. This may well require a more rigorous and less flexible approach to class scheduling, with clear segregation of transactions and region classes.

10.2.3 Wait-for-Input Transactions

Transactions can be defined as wait-for-input by specifying the WFI parameter on the TRANSACT macro in the IMS Stage 1. A WFI transaction is scheduled in a region when the first message arrives and remains resident in that region until the processing limit count is reached, the region is stopped by the MTO, or the IMS system is shut down. When one of these is the case, IMS returns a QC status to the next message GU call.

Otherwise, when the program issues a message GU, IMS either returns the next message to the program, if there is one, or else suspends the application until a message does arrive in the input queue for that transaction.

**Warning**

If this wait time is greater than the SMF wait time, the message region will abend with a S522 abend. This can be avoided either by putting the message region's job name in the MVS PPT table with the “no timing” option or by implementing an SMF exit to prevent cancelling the region if the wait time expires.

For a WFI transaction, there should be less queueing and minimal scheduling cost because program load and initialization are eliminated. Storage isolation should be considered for these transactions since the dependent region can have very low occupancy when the input transaction rate is reduced.

A WFI transaction should be the only transaction assigned to the associated program, since no other transaction can execute in that region. In theory, another transaction could be defined for the program, and the program could be defined as eligible for parallel processing. We do not recommend this.

To support very high transaction arrival rates, it may be necessary to implement parallel scheduling for a WFI transaction. On the TRANSACT macro, set PARLIM=1 and set MAXRGN equal to the maximum number of WFI regions required (see 10.2.6, “Parallel Scheduling” on page 212).

WFI is normally thought of as a way of handling important, high-volume transactions. However, it should also be considered if you have transactions that perform a large amount of initialization processing each time they are scheduled.
10.2.4 Pseudo-WFI Transactions

If the overall transaction arrival rates are high, but an inconsistent mix of transaction codes is used, then WFI is unlikely to be of great benefit. In this case, pseudo-WFI (PWFI) might provide significant benefit. It effectively provides a dynamic WFI environment that allows WFI to be switched on for any transactions that are temporarily able to benefit from the reduced costs of WFI processing.

It is requested by specifying PWFI=Y as an execution parameter in the dependent region JCL. Any transaction defined with MODE=SNGL and with PROCLIM > 0 running in that dependent region can exploit pseudo-WFI.

On a message GU call, if there are no more messages for a PWFI transaction, IMS does not immediately return a QC status code. IMS first checks to see whether other transactions are on the queue that can be processed in that region. If an eligible transaction is waiting on the queue, then a QC is returned to the message GU, the program terminates, and the waiting transaction is scheduled into the region. However, if no other work is available and the resources owned by the region are not required by another region (in other words, there are no pool shortages), then the program is suspended. It waits on its message GU (as for WFI) until a new transaction arrives that can run in the region. If the next transaction is the same as the suspended transaction, then the message is passed to the waiting GU and the benefits of WFI are realized. If a different transaction arrives, then a QC is returned to the waiting message GU, the program terminates and the new transaction is scheduled into the region. If the new message is for the transaction that is scheduled in the region, IMS returns the new message to the application program terminating and rescheduling the resources.

A consequence of pseudo-WFI is that all the PWFI regions become occupied, even at very low transaction volumes. This means that IMS must have enough space in all its pools (PSB, PSBW, and other pools) to hold resources for all the PWFI regions concurrently.

The cost of pseudo-WFI is very small. If a transaction arrives in the system and the required program is not waiting in a PWFI region, then scheduling of the new transaction includes the terminating of one of the waiting PWFI programs. However this is usually very quick and unlikely to be perceived by terminal users.

The potential performance benefits of pseudo-WFI are quite significant, and the use of pseudo-WFI is recommended.

The effects of PWFI show up only on the IMS monitor program I/O report. An example is shown in Figure 38 on page 107. There is one report line against the I/O PCB for each time that the program is scheduled. The DDN/FUNC value is **W F I. The reported data indicates how many transactions were processed for the one scheduling and the total, mean, and maximum times spent waiting between transactions for that scheduling. The larger the number of transactions processed per scheduling, the greater the benefit.
10.2.5 Process Limit Count and IMS Quick Rescheduling

IMS uses process limit count to ensure that no transaction type can monopolize a message region if other transactions are waiting and are eligible for processing in that region. Quick reschedule allows application programs to process more than the processing limit of messages for each physical schedule. It eliminates processing overhead caused by unnecessary rescheduling and reloading of application programs. We discuss performance related issues here.

10.2.5.1 PROCLIM

The process limit count (PROCLIM or PLC) of a transaction specifies how many waiting messages can be processed after the program has been scheduled and before IMS assesses whether it should be allowed to continue (quick reschedule). PROCLIM has relevance only when transactions arrive faster than they are processed, so that a queue of waiting messages builds up.

If PROCLIM=0, one and only one message is processed per program scheduling (no quick reschedule and no pseudo-WFI). 15

If PROCLIM=65565, the number of messages that can be processed per scheduling is unlimited.

10.2.5.2 Quick Reschedule

Quick reschedule is not an option. It is always enabled for transactions with a PROCLIM greater than zero. It allows application programs to process more than the PROCLIM number of messages per schedule. Quick reschedule eliminates the processing overhead of rescheduling and reloading application programs.

A region exploits quick reschedule only when:

- No other work of equal or higher priority is available for the region to process.
- The same transaction would be scheduled if the region terminates and reschedules.
- The region is an MPP processing a MODE=SNGL transaction.
- The processing limit count is greater than zero.
- The PSB is not allocated with the dynamic PSB option (DOPT).

When a region goes through quick reschedule, the message count that is compared with the processing limit count is reset, the accounting (X’07’) and scheduling (X’08’) log records are written, and the next message is returned to the application for processing.

Flags in the accounting and scheduling records written during a quick reschedule indicate that the records do not include actual program termination and scheduling times. These records are written for accounting purposes only. Restart and backout do not use these records.

---

15 In IMS 3.1, specifying PROCLIM=1 was a means of avoiding quick reschedule. However, starting in IMS 4.1, this is no longer the case.
10.2.5.3 Choosing PROCLIM Values
Traditionally, lower-priority transactions were given low PROCLIMs, and higher-priority transactions were given higher PROCLIMs. This was done to prevent low-priority transactions from monopolizing dependent regions. Some IMS customers who have a broad variety of arriving transactions and who guarantee to have enough regions and CPU power to cope with whatever transaction demands are placed on them, have a very simple class structure and let all PROCLIMs default to 65565.

Because of quick reschedule, it is now practical to set all PROCLIMs relatively low (in the range of 2 to 10):
- High-priority and high-volume transactions repeatedly exploit quick reschedule.
- Lower-priority and high-volume transactions gain the performance advantages of quick reschedule when no competing higher-priority transactions are present.
- High-priority but low-volume transactions do not have to wait long if lower priority transactions are active.

All the same, the highest-priority high volume transactions benefit from using a little less CPU and writing fewer log records (types X'07' and X'08') if a larger PROCLIM is specified.

10.2.5.4 PROCLIM with WFI Transactions
For WFI transactions, set the process limit count reasonably high so as not to incur the scheduling overhead, but low enough to capture the X'07' log record for accounting and performance monitoring. A reasonable PLC for a WFI transaction is in the range of 800 to 1000 transactions. Your final decision must be based on your system.

10.2.6 Parallel Scheduling
Do not use parallel scheduling for low-volume transactions that can be adequately scheduled in one region. Reserve parallel scheduling for those transactions that arrive at such a rate that they tend to queue on the message queue. A transaction that has a temporary arrival rate exceeding the ability of a message region to handle the processing of that transaction is also a good candidate for parallel scheduling if the temporary high arrival rate is periodic during the day. If the arrival rate is high only once for a very short time, the transaction is not a good candidate for parallel scheduling.

Parallel scheduled transactions can degrade response time of other transactions if they occupy more than one message processing region and tend to execute for a long time. Therefore, consider the use of parallel scheduling carefully.

Parallel scheduling is defined by the MAXRGN and PARLIM keywords on the TRANSACT macro and can be changed during system operation by the /ASSIGN command for PARLIM and /CHANGE command for MAXRGN:
- MAXRGN—limits the number of MPP regions that can be concurrently scheduled to process a transaction.
- PARLIM—an additional region is scheduled (subject to the MAXRGN limit) whenever the current transaction enqueue count exceeds the PARLIM value multiplied by the number of regions currently running the transaction.
10.3 Scheduling Failures

Having selected a target transaction to be processed in an available region, IMS then checks for conflicts that may prevent the transaction from being processed.

10.3.1 External Conflicts

The following are the main external conflicts:

- **Program Conflict** - Transaction or PSB is stopped or locked. The program is already scheduled in another region and parallel scheduling is not allowed. This means it is defined with (or has defaulted to) SCHDTYPE=SERIAL on the APPLCTN macro. There are two possibilities:
  - Transaction is not eligible for further parallel scheduling. Occurs because the transaction is already scheduled and has PARLIM=0, or PARLIM/MAXRGN prevents additional scheduling.
  - A Fast Path database, referenced in the PSB, is not available.

- **Priority Conflict** - The IMS scheduler has already tried to schedule work into the region, but it failed because of an internal conflict (see the following). The scheduler has tried to find an alternative transaction to schedule in the region, but has failed on this attempt because of the SCHD option of the original transaction.

- Miscellaneous other reasons cause external conflicts:
  - PSB could not be found or read from ACBLIB.
  - PSB includes a database PCB with PROCOPT=L.

In the event of an external conflict, the IMS scheduler looks for another transaction to attempt to schedule, or waits if none are available. Otherwise, if no external conflict exists, IMS checks for an internal conflict, sometimes misleadingly referred to as an intent conflict.

10.3.2 Internal Conflicts

There are just two possibilities for internal conflicts (or intent conflicts).

- This or an already scheduled program has PROCOPT=E on a database or segment common to both PSBs.

- Insufficient space is available in the PSB, PSBW, EPCB (for Fast Path), or DMB pool to satisfy this scheduling request.

In the event of an internal conflict, the IMS scheduler attempts to find an alternative transaction to schedule but is constrained by the SCHD parameter on the TRANSACT macro of the original transaction that suffered the internal conflict. The parameters are these:

- **SCHD=1** → allow equal or higher priority transaction of same class
- **SCHD=2** → allow only higher priority transaction of same class
- **SCHD=3** → allow any transaction of same class
- **SCHD=4** → try another class

A failure to meet the SCHD criteria causes a priority-cutoff external conflict and causes IMS to look for yet another transaction for the waiting dependent region.
However, after five successive scheduling failures result from priority cutoff, the SCHD is ignored and the next region class is automatically considered for scheduling (as though using SCHD=4).

### 10.4 Scheduling Completion

When no conflicts are present, IMS acquires the PSB and if necessary, any missing DMBs. This requires the services of the block mover, whose responsibilities are:

- Issue BLDL if DOPT PSB.
- Clone, or read and relocate, the PSB.
- Read any DMBs that are not resident or in the DMB pool.
- Obtain the PSB work area in the PSBW pool.

Two types of IWAITs can therefore be possible at this time:

- I/O IWAIT to ACBLIB
- Non-I/O IWAIT when block builder is busy servicing another region. With the new latch manager introduced in IMS/ESA V4.1, different block mover functions each have their own latches. Consequently, it is less likely that a request has to wait because the block mover is busy.

Once all the blocks are available and allocated, scheduling is complete.

### 10.5 Region States

A dependent region is either occupied or idle. In each case, the region can be in three possible states.

#### 10.5.1 Region Idle

When a region is idle, the three possible states are:

1. **No messages** — If any messages are waiting, none has the same class as the region (non-I/O IWAIT).

2. **Waiting for the block mover** — The block mover is busy servicing another region (non-I/O IWAIT).

3. **Idle for intent** — Do not confuse this with intent conflict. *Idle for intent* means that the region is available and that at least one message on the input queue could be processed in that region. However, for several possible reasons, the scheduler is unable to schedule any work in the region. Examples of what can cause this are:

   - The only transactions eligible cannot be scheduled because their programs are already scheduled and have SCHDTYP=SERIAL.
   - A waiting transaction is parallel schedulable, but no further copy of the program is allowed because of the MAXRGN or PARLIM parameter.
   - The first-choice transaction has failed to schedule as a result of an internal conflict (PROCOPT=E or pool space failure).
   - The IMS scheduler has found other candidate transactions, but scheduling has failed with priority cutoff because of the SCHD option.
Idle for intent is not necessarily all IWAIT time. The scheduler attempts to schedule new transaction and, if that fails, the region enters another non-I/O IWAIT.

10.5.2 Region Occupied
When a region is occupied, the three possible states are:

1. **Scheduling**
   - The IMS scheduler is executing, for example, cloning a PSB (non-IWAIT time).
   - The IMS scheduler is waiting on ACBLIB I/O (IWAIT time).

2. **Schedule end to first DL/I call**
   - A program is loading.
   - A program is initializing.

3. **Program Execution**

10.5.3 IMS Monitor Reports
The IMS Monitor reports on the different states of a region.

10.5.3.1 Region Summary Report
- “Scheduling and Termination” section — Reports on
  - NOT-IWAIT: the scheduler is executing on behalf of the region, which is now considered occupied
  - Elapsed: the scheduler is executing or waiting for I/O to ACBLIB, and the associated region is considered occupied.

- “Schedule to First Call” section — Reports on program load, program initialization, and so on, all seen as non-IWAIT time.

- “Elapsed Execution” section — Program elapsed time from first DC call to termination.

- “Idle for Intent” section — The time when region is in idle-for-intent state. The NOT-IWAIT time indicates how much time the scheduler spent executing in an attempt to schedule transactions into this region.

10.5.3.2 Region IWAIT Report
- “Scheduling + Termination” section — Reports on
  - NO MESSAGES Non-I/O IWAIT when no messages are eligible for this region.
  - BLR Non-I/O IWAIT when block mover is busy.
  - INTENT Non-I/O IWAIT when region is idle for intent. This is the wait time component of the Idle-for-Intent time reported on the region summary report. It relates to individual failures, whereas the Region Summary treats the whole period of idle for intent as a single entity.
  - INT I/O IWAIT to read an intent list from ACBLIB
- PSB  I/O IWAIT to read a PSB from ACBLIB
- DMB  I/O IWAIT to read a DMB from ACBLIB
- DD=SHMSG  Short message queue I/O IWAIT time getting the input message
- DD=LGMSG  Long message queue I/O IWAIT time getting the input message

- “DL/I Calls” section — Reports on
  - DD=database name  I/O IWAIT for database access
  - PI=segname...segcode  Non-I/O IWAIT for Program Isolation

An example is shown in Figure 70 on page 217.

10.6 Monitoring Scheduling Failures

10.3, “Scheduling Failures” on page 213 discusses how scheduling can fail for external and internal (intent) conflicts. In summary, an internal or intent conflict is the result of PROCOPT=E or a pool shortage. Anything else is an external conflict.

Information about these conflicts (scheduling failures) is written to the log and the IMS monitor.

10.6.1 IMS Monitor

The IMS monitor makes a variety of reports available.

10.6.1.1 Reports Report

This reports individually on the numbers of both types of internal conflicts (see Figure 71 on page 218).

Intent failure refers only to problems associated with the use of PROCOPT=E.

In the case of scheduling pool failures, the pool identifiers used are:

<table>
<thead>
<tr>
<th>Pool ID</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLMP</td>
<td>ECSA PSB pool</td>
</tr>
<tr>
<td>DSPP</td>
<td>DL/I SAS PSB pool</td>
</tr>
<tr>
<td>PSBW</td>
<td>PSB work pool</td>
</tr>
<tr>
<td>DLPD</td>
<td>DMB pool</td>
</tr>
<tr>
<td>EPCB</td>
<td>EPCB pool</td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>OCCURRENCES</strong></td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>SCHEDULING + TERMINATION</strong></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>690675963</td>
</tr>
<tr>
<td><strong>...SUBTOTAL...</strong></td>
<td>203</td>
</tr>
<tr>
<td>3</td>
<td>83743</td>
</tr>
<tr>
<td>2</td>
<td>76072</td>
</tr>
<tr>
<td><strong>..TOTAL...</strong></td>
<td>208</td>
</tr>
<tr>
<td><strong>DL/I CALLS</strong></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>1610641</td>
</tr>
<tr>
<td>1448</td>
<td>6977199</td>
</tr>
<tr>
<td>107</td>
<td>335362</td>
</tr>
<tr>
<td>311</td>
<td>1153056</td>
</tr>
<tr>
<td>740</td>
<td>4691266</td>
</tr>
<tr>
<td>143</td>
<td>679268</td>
</tr>
<tr>
<td>722</td>
<td>5264044</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>93005</td>
</tr>
<tr>
<td>58</td>
<td>772925</td>
</tr>
<tr>
<td>7</td>
<td>34996</td>
</tr>
<tr>
<td>2</td>
<td>264191</td>
</tr>
<tr>
<td>6</td>
<td>149706</td>
</tr>
<tr>
<td>3</td>
<td>17940</td>
</tr>
<tr>
<td><strong>..TOTAL...</strong></td>
<td>7756</td>
</tr>
</tbody>
</table>

Figure 70. IMS Monitor REGION IWAIT Report (Showing Scheduling IWAITs)
10.6.1.2 Other Monitor Reports

10.5.3, “IMS Monitor Reports” on page 215 explains how the region summary and region IWAIT reports show the impact of idle for intent when a region is free and eligible messages are waiting but cannot be scheduled. This is further illustrated by Figure 72 on page 219, which shows a composite set of reports from the monitor relating to a system with significant use of SCHDTYP=SERIAL and tightly constrained parallel scheduling.

10.6.2 IMSPARS

IMSPARS, in its “Internal Resource Usage—Scheduling Statistics” report, shows the number of scheduling failures. The events itemized on the report are:

- “Program Conflicts” (resulting from SCHDTYP=SERIAL)
- “Database Intent Conflicts” (incorrectly named internal or intent conflicts)
- “Conflicts for Misc Reasons” (all other types of external conflict)

The report of scheduling failures is shown in Figure 73 on page 220.
**Chapter 10. Scheduling and Dependent Region Considerations**

**IMS MONITOR ****REGION SUMMARY**** TRACE START 1996 028 11:00:02 TRACE STOP 1996 028 11:13:00 PAGE 0025**

<table>
<thead>
<tr>
<th>IDLE FOR INTENT</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 27</td>
<td>17</td>
<td>10321132</td>
<td>607125</td>
<td>1254602</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 28</td>
<td>11</td>
<td>5853596</td>
<td>532145</td>
<td>964979</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 29</td>
<td>20</td>
<td>11508591</td>
<td>575429</td>
<td>1067160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 30</td>
<td>14</td>
<td>7672905</td>
<td>548064</td>
<td>1141397</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IMS MONITOR ****REGION IWAIT**** TRACE START 1996 028 11:00:02 TRACE STOP 1996 028 11:13:00 PAGE 0050**

<table>
<thead>
<tr>
<th>SCHEDULING + TERMINATION</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>FUNCTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 29</td>
<td>982</td>
<td>660052496</td>
<td>672151</td>
<td>2171280</td>
<td>NO MESSAGES</td>
<td>MSC</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>1108578</td>
<td>25194</td>
<td>147187</td>
<td>INTENT</td>
<td>MSC</td>
</tr>
<tr>
<td><strong>SUB-TOTAL</strong></td>
<td>1026</td>
<td>661161074</td>
<td>644406</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IMS MONITOR ****REPORTS**** TRACE START 1996 028 11:00:02 TRACE STOP 1996 028 11:13:00 PAGE 0170**

- NO INTENT FAILURES IN THIS TRACE
- NO POOL SPACE FAILURES IN THIS TRACE
- NO DEADLOCKS IN THIS TRACE
- TOTAL TIMES ECBs WAITED FOR SAPS = 0
- MONITOR OVERHEAD DATA

Figure 72. IMS Monitor Reports for Program Conflicts
**IMS INTERNAL RESOURCE USAGE**

**SCHEDULING STATISTICS**

<table>
<thead>
<tr>
<th></th>
<th>COUNT</th>
<th>/TRANSACT</th>
<th>/SECOND</th>
<th>% OF ALL CONFLICTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM CONFLICTS</td>
<td>449</td>
<td>.01</td>
<td>.32</td>
<td>.94%</td>
</tr>
<tr>
<td>DATABASE INTENT CONFLICTS</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>CONFLICTS FOR MISC REASONS</td>
<td>47,556</td>
<td>1.33</td>
<td>33.53</td>
<td>99.06%</td>
</tr>
</tbody>
</table>

**SMBS TRIED FOR SCHEDULING**

- 74,238, 2.08, 52.34

**PROGRAMS SCHEDULED**

- 26,233, .73, 18.49

**TOTAL CONFLICTS**

- 48,005, 1.34, 33.84, 64.66% OF SMBS TRIED

*Figure 73. IMSPARS Internal Resource Usage Scheduling Statistics*
10.6.3 Summary of Terminology

Unfortunately, the terminology surrounding scheduling issues can be very confusing. The following table illustrates the different terminology used in different places:

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>intent conflict</td>
<td>Internal conflict (pool space failure or PROCOPT=E).</td>
</tr>
<tr>
<td>database intent conflict</td>
<td>IMSPARS name for internal conflict (as above).</td>
</tr>
<tr>
<td>intent failure</td>
<td>IMS monitor reports report: failures due to PROCOPT=E.</td>
</tr>
<tr>
<td>idle for intent</td>
<td>The state of an idle region when eligible waiting transactions cannot be scheduled for any reason. The elapsed time is reported on the IMS monitor region summary report.</td>
</tr>
<tr>
<td>intent IWAIT</td>
<td>A non-I/O IWAIT during the period a region is idle for intent. The wait time is reported on the IMS monitor region IWAIT report.</td>
</tr>
</tbody>
</table>

10.7 Schedule End to First DL/I Call

Schedule End to First DL/I Call is the elapsed time from schedule approval until the first DL/I call (message GU) from the application program. The processes that occur within this period are outside the control of IMS. After a program has been approved for scheduling, IMS notifies (through interregion communication) the dependent region as to which program to schedule. The dependent region is then dispatched by the operating system when higher priority tasks (including IMS control regions) have no work to do.

The dependent region program controller (DFSPCC20) then performs the program load function.

After it is loaded, the program receives control and usually performs housekeeping or initialization functions and then issues a GU to the I/O PCB. At this point, the schedule-end-to-first-call elapsed time is completed.

10.8 Program Load Options

In the past, the loading of the IMS application program sometimes accounted for a major part of the transaction internal response time. Even though a program library had a large block size, the average size of the blocks was often considerably smaller than the maximum possible. This resulted when small blocks of relocation control information, for example, were included with the program text blocks. Therefore, program load can require many I/Os. Also, in many cases the amount of data read in (in other words, the program) was many times the amount of data read by the program from the databases. Program load times of 0.25 second were typical, and load times measured in seconds were not unusual.

Today, program load time need no longer be a major issue, and in fact, four different techniques are available to IMS:

- Standard program fetch (LINK)
10.8.1 Standard Program Fetch

IMS checks the dependent region BLDL list (see 10.8.1.2, “Dependent Region BLDL Option” on page 223) to determine if the DASD address of the required module is available. If so, IMS issues a LINK SVC pointing to the location of the program on external storage, and MVS does not have to issue a BLDL to find the program. If IMS does not find the program name in the dynamic BLDL table, it issues a BLDL for that program, and the name and pointer to the program are saved in the table. IMS then issues an OS LINK using the pointer just found, and the MVS/ESA program fetch routine is invoked to find the application program.

Normally, application programs in the 80 KB to 120 KB range load in approximately 150 ms to 200 ms. If the load times are greater than the guideline, perform additional analysis on those transactions or regions.

10.8.1.1 Program Load Libraries

A major factor in standard fetch elapsed time is the search order of the program libraries. STEPLIB and JOBLIB are the first program libraries searched, if they are present. If they are not present, place the IMS program library (IMSESA.PGMLIB) first in the LNKLSTxx member of SYS1.PARMLIB (SYS1.LINKLIB is always searched first). If IMSESA.PGMLIB is placed in the LNKLST of MVS, it is managed by the LLA address space. This can reduce program load time. Refer to Section 10.8, “Program Load Options” on page 221 for a more detailed explanation of the LLA address space.

It is recommended that you place IMSESA.PGMLIB first in the STEPLIB/JOBLIB DD statement list of an MPP region, because RESLIB is only accessed at IMS initialization, and abend processing and PGMLIB could be referenced for every program scheduled.

In BMP regions, place the RESLIB DD statement first because approximately 12 modules are loaded from this library for region initialization and usually only one module is loaded from the application program library. Search time and load time are two critical factors in schedule end to first DL/I call elapsed time for low-volume, nonpreloaded programs.
The search order for MVS program libraries is:

1. JOBPACK
2. STEPLIB or JOBLIB
3. IEAFIXxx list
4. IEALPAxx list
5. Active PLPA CDE list
6. Hash technique of LPA directory
7. MVS LLA address space/LNKLST libraries.

10.8.1.2 Dependent Region BLDL Option

The BLDL list is a hash table containing the locations of:

- Preloaded modules
- Virtual fetch modules
- Standard fetch modules (the most recently referenced)

The default BLDL list in the dependent region contains space for 20 addresses of modules accessed by standard fetch. The dependent region execution parameter DBLDL allows values between 0 and 9999 to be specified. Standard fetch programs that have a high usage and are not preloaded are likely to have an entry in the list.

The purpose is to save the addresses of the most recently used message processing programs. Every time a program with an entry in the list is loaded, the reference count in the table entry is incremented by one. When a program not in the table is loaded, the reference counts of all entries in the table are decremented by one and the entry with the lowest count is replaced with information about the new program.

Programs with entries in the BLDL list should have an elapsed time for schedule-end-to-first-DL/I-call of less than 250 ms. This is true as long as the application programs are not more than 200 KB. Make the number of BLDL entries for each region large enough to hold the entries for all application programs that can execute in that region.

Note

If application programs are changed in IMSESA.PGMLIB, the message regions must be stopped and restarted to get the correct directory entry in the BLDL list.

If programs are being tested in a test region, specify the number of dynamic BLDL entries on the MPR EXEC statement as zero. This bypasses the dynamic BLDL code and always loads the program from the external program library. The specification enables a test program to be link-edited again, and the new copy of the program is executed on the next test of the transaction.
10.8.2 Preload

Application programs and subroutines can be loaded into a dependent region when it starts up. These are the preloaded programs. The programs to be preloaded must be listed in a DFSMPLxx member of IMSESA.PROCLIB, and the dependent region JCL includes a parameter indicating the appropriate suffix (PRLD=). The preload list can also reference modules loaded in the MVS link pack (PLPA).

When the dependent region is started, the sequence of events is as follows:

1. The region controller task:
   - Reads the preload list:
   - Builds tables for preload (and virtual fetch) addresses.
   - Gets storage for the BLDL table (calculated from the JCL DBLDL parameter).
   - Issues BLDL for preloaded modules.
   - Loads reentrant preload modules.
   - Loads preload modules not found by BLDL.
   - Stores all module addresses in the BLDL table.
   - Attaches the program controller task.

2. The program controller task:
   - Loads reentrant modules again. If the controller task collapses as a result of a program abend, these modules remain resident.
   - Initializes virtual fetch.
   - Loads nonreentrant modules and adds their addresses to the BLDL table. The modules are reloaded when the program controller is reattached.

When a preloaded program is scheduled, control is passed to it simply by branching to its address. When subroutines are called by application programs, preloaded modules are found in the JOBPACK CDE list. Therefore, directory search or program load I/O activity is not incurred, only the path length of LOAD.

Preloaded message processing programs do not require the link-editor attributes of RENT or REUS (though they must, of course, be written to be serially reusable). However, all preloaded subroutines must have either the RENT or REUS attribute.

If a preloaded program has an alias that is invoked, the alias must also be included in the preload list.

If any abend occurs in a dependent region, either a real abend or pseudo-abend such as a 0Cx or U777, all nonreentrant programs that were preloaded appear on the region dump. Also, reattachment of the program controller (DFSPCC20) takes longer because all nonreentrant programs that were previously preloaded are automatically preloaded again.

Several application program characteristics can affect function and performance under the preload feature:

- To change a program that has been preloaded in a dependent region, the program must be recompiled and relinked into the program library specified in
the message region. While this is being done, IMS continues to invoke the old copy of the program. Once the program has been replaced, the region must be stopped and restarted in order to load the new copy.

- Protection for abending application programs is provided by IMS (not by MVS/ESA). Any program that abends is stopped by IMS in the program directory (PDIR). (The transaction type active at the time of the abend is also stopped.) It is therefore possible to issue a /START for an application program that has abended. The effect is reactivation of a copy of a program that previously abended.

- A fresh copy of preloaded application programs can be obtained only by stopping and restarting the regions containing the preloaded module after a new copy of the program has been link-edited into the program library. Preloaded modules that actually reside in link pack or fixed link pack can be refreshed only by another IPL of MVS and doing a CLPA. The CLPA is required only if the changed module resides in PLPA. LPA modules can be overridden by specifying a JOBLIB or STEPLIB DD statement in the message-region procedure that points to the library containing the module to be referenced.

- In a preloaded environment, program initialization becomes very important. We recommend front-end initialization application programs so initialization occurs immediately upon entry into an application or immediately prior to issuing a GU call to the message queue (Figure 74).

![Example: Program Initialization Structure](image)

Figure 74. Program Initialization Structure

- The coresidence of multiple application programs within the same address space might be considered an integrity exposure. Programs can be addressed to an entire address space and may modify an area (with problem-program protect key) within that address space. Reentrant modules from authorized libraries are loaded into key zero storage in MVS (subpool 252) and therefore must not modify themselves. If they do modify themselves, an 0C4 abend occurs.

Guidelines for the most effective implementation of program preload include these:

- Preload all commonly used PL/I or COBOL subroutines (or common runtime routines for PL/I or COBOL of the language environment) and application program subroutines into each dependent region. If these subroutines are reentrant, put them into PLPA as well, so that only one copy resides in virtual storage. If they are put into PLPA or MLPA, remove them from the STEPLIB libraries so that the copy in the PLPA is used. The subroutines that are loaded
can be traced with GTF. The output from this report can be used to match the programs being loaded with the program names in the region preload list to ensure that the appropriate modules are preloaded.

• Application programs are candidates for preload when they account for a high percentage of a region schedule volume. Preload is most effective when the transaction arrival rate for the preloaded program is adequate to keep the working set of the program in real storage. If a system is constrained by real storage contention, preload subroutines only; preloading application programs increases the paging rate. (It is not always true that paging I/O is more efficient than program fetch.) Program elapsed times for preloaded programs also tend to be longer as a result of page faulting through the application code when the working set is not in real storage.

Consider class scheduling of high-volume transactions along with the decision to preload. Depending on the transaction arrival rate, it may be advantageous to preload in only one or two regions and class schedule the transactions accordingly.

WFI is another way to schedule a high-volume transaction. If a WFI region or a dedicated region with one or two preloaded programs is specified, consider using the storage isolation function.

The use of the preload option for system performance is highly dependent on the availability of real storage and the arrival rate of the candidate transactions.

• Application program overlay is sometimes a viable alternative to preload. If an all-purpose application program is coded to interrogate the input transaction data and execute only a portion of its code for the transaction subcode, program overlay I/O may be more efficient than loading or paging through the entire program. Preload is a better alternative if the transaction arrival rate and real storage are sufficient to maintain a working set.

• Extremely large application programs that have a relatively small working set are also candidates for preload. The load time for these programs could be in the one to two second time frame and cause a high paging rate when they are loaded. If they are preloaded, they can sustain a few page faults and not disrupt the rest of the system.

10.8.3 Library Lookaside/Virtual Lookaside Facility

Library lookaside (LLA) maintains and controls in-storage copies of directories of production libraries. It also stages selected modules to a virtual lookaside facility (VLF) data space. In VLF, LLA maintains a set of modules that have been determined to have the optimum characteristics for being kept in storage. LLA function provides significant improvements over previous methods of loading modules.

VLF provides services that enable data objects to be maintained in virtual storage rather than repeatedly retrieved from auxiliary devices. VLF services can improve performance and response time whenever a single user or several concurrent users seek repeated access to the same data object as it avoids the I/O for loading.

LLA can also use VLF to save and retrieve frequently used modules to improve IMS response time and reduce I/Os. Thus, LLA can provide significant performance improvement, particularly faster response time for end users, by reducing repeated retrievals of objects from partitioned data sets. Also, LLA can be
used effectively to improve the IMS response time by specifying the names of IMS application program libraries in the CSVLLAxx member of SYS1.PARMLIB.

Figure 75 on page 228 shows the relationships between the IMS-dependent region, the LLA address space, and the object that represents the application load module.

### 10.8.3.1 Setting up LLA/VLF for IMS

In the COFVLFxx member of SYS1.PARMLIB, specify the class and the major name for the LLA address space as follows:

```
CLASS NAME(CSVLLA)
EMAJ(LLA)
MAXVIRT(nnn)
```

where nnn is the maximum storage to be used for holding application programs in the data space.

In the CSVLLAxx member of SYS1.PARMLIB, specify the names of the IMS libraries you intend to have LLA/VLF manage, as follows:

```
LIBRARIES(libname1,libname2,...)
and/or
(-LNKLST-)
FREEZE
```

The FREEZE is necessary to keep the directory entries for the libraries in LLA virtual storage. (LNKLST libraries are always processed as FREEZE.) LLA/VLF works best if the libraries contain heavily used and seldom updated read-only modules.

Start the LLA address space by using the following system command:

```
S LLA,SUB=MSTR,LLA=xx
```

You also could change the LLA dynamically by issuing the following command:

```
F LLA,UPDATE=xx
```

where xx denotes the suffix of a CSVLLAxx member of SYS1.PARMLIB.
To update the directory information for all libraries that LLA manages, use the following command:

F LLA,REFRESH

10.8.3.2 Dependent Region DBLDL Parameter and LLA
LLA with the FREEZE option keeps the program library directories in storage and also processes BLDL requests. The dependent region does not need to also save module addresses. Additionally, the LLA directory does not need to be refreshed if the dependent region still has the old addresses in its BLDL table.

The recommended options are:

FREEZE and DBDL=0

10.8.3.3 LLA/VLF Tuning Considerations
As the services are invoked using the cross-memory program control function of MVS/ESA, the DPRTY of the VLF address space can be set a little lower than that of the IMS control region.

Use storage isolation for the VLF/LLA address spaces wherever possible, as this protects the pages.

In the COFVLFxx member of SYS1.PARMLIB, specify MAXVIRT to limit the amount of virtual storage used for each VLF class. MAXVIRT is used to control the amount of data kept in each data space. If it is set too low, it causes additional I/Os of data within the data space due to forced trim; if it is set too high, it causes too little used data to be paged out due to lack of use.
LLA evaluates which programs to save in the VLF data space based on sampling 2000 fetches. If the number of used programs is more than 666, then the sample size is too small. Use the CSVLLIX1 user exit to force sampling sizes at least three times the number of used programs in the program library.

SMF records (type 41 subtype 3) are written every 15 minutes and give VLF statistics on searches and loads. Use these statistics to adjust the MAXVIRT size, taking care that real storage is not impacted too much, causing excessive I/Os to the page data sets.

10.8.4 Virtual Fetch

Virtual fetch is usually less powerful in one way or another than LLA/VLF.

Virtual fetch uses the facilities of MVS to reduce the time and system resources required for loading application programs in the IMS dependent regions. Instead of loading programs using standard fetch (LINK), it uses VIO paging, which can be more efficient. However, virtual fetch does not support the LOAD function of program fetch and therefore cannot be used to manage any subroutines or subprograms. It also does not support any load modules link-edited with the overlay option.

Improved performance of the IMS system is achieved because the path length to load application programs through virtual fetch is shorter than for standard fetch. The elapsed time is also much shorter because the program is block-paged-in rather than program-fetched. This improvement can be achieved if an adequate paging subsystem is defined to the MVS system.

Virtual fetch is a nonswappable started task that runs in its own address space. There is only one virtual fetch task in an MVS system:

```
EXAMPLE OF STARTED PROCEDURE FOR VIRTUAL FETCH ADDRESS SPACE:

//IEFPROC EXEC PGM=CSFVVFCRE,TIME=1440
//VF0N00 DD DSN=LIB1,DISP=SHR
//VF0N01 DD DSN=LIB2,DISP=SHR
//VF0N02 DD DSN=LIB3,DISP=SHR
//SYSUDUMP DD SYSOUT=A
/*

At initialization time, this task is given a list of program libraries that virtual fetch is to manage. The program libraries are defined to virtual fetch by DD statements with the name of VF0Nxx. The xx is a number from 00 to 99, and each VF0Nxx is processed in numerical order by virtual fetch. These libraries cannot be concatenated. Virtual fetch reformats all of the programs in the libraries specified and places a relocatable copy of the programs in page format (4 KB blocks) on the VIO page data sets.

Copy the programs that virtual fetch manages into a special library to be used by the virtual fetch address space. This copying reduces the amount of external DASD storage on the VIO paging volumes that are required to contain the reformatted load modules from the program libraries. Virtual fetch reformats and pages out all of the programs that reside in the libraries specified to it.
Virtual fetch can manage only one copy of a program; therefore, test programs with the same name cannot be managed with virtual fetch.

The contiguous slot algorithm of MVS/ESA is used in writing these pages to the page data sets. This reduces the number of seeks and rotations required when loading the programs. The contiguous slot allocation causes the pages to be written out in bursts and spread across all of the VIO paging data sets. This allows the program to be loaded into virtual storage using concurrent I/O requests when possible.

When virtual fetch initialization is complete, the task goes into a wait state and no additional processing is done under this TCB unless a virtual fetch address space refresh is requested.

10.8.4.1 IMS Specification of Virtual Fetch
There are two subparameters on the IMS message region procedure for the use of virtual fetch.

The first one is the VSFX subparameter. This subparameter specifies the suffix of the DFSVFLxx member of IMSESA.PROCLIB where the names of the programs to be managed by virtual fetch are specified. This list can be a subset of the programs that are managed by virtual fetch.

The second subparameter, VFREE, specifies whether the storage allocated in the message regions to hold the programs is to be handled as FREEMAIN. If the NO FREEMAIN option is chosen (VFREE=N) virtual fetch issues a PGRLSE (page release) macro which releases the real and auxiliary space associated with the module. This frees these resources, while keeping the virtual storage allocated in the message region. If a virtual fetch GETMAIN command fails, virtual fetch frees all the virtual storage acquired to hold the programs and starts over. This option is not recommended because if a non-virtual-fetch-managed program is executing and requests additional virtual storage via a GETMAIN macro and not enough free virtual storage is available to honor the request, the program requesting the storage abends with an 80A abend code. This is not good for IMS performance.

The option of VFREE=Y specifies that the virtual storage needed to hold the program is to be dealt with as FREEMAIN at program termination. This helps to eliminate the possibility of the 80A abend described. We recommended the VFREE=Y option if virtual fetch is implemented.

10.8.4.2 IMS Use of Virtual Fetch
When a program is required, the IMS message region program controller accesses the region hash table to determine if the program to be executed is known (in other words, preloaded, available for virtual fetch, or recently fetched from the program library). If the requested program is in the preload table, a direct branch to the program is taken. If it is managed by virtual fetch, virtual fetch is called to invoke the application program.

If an error is encountered by virtual fetch in loading the program, a nonzero return code is returned to IMS and standard fetch (LINK) is used to invoke the application program. If an I/O error occurs on one of the VIO page data sets, it can be cleared by refreshing virtual fetch.
The I/O to load the application program is executed under the TCB of the dependent region. The program is block paged into the dependent region from the VIO data sets with concurrent I/Os. Control is then passed to the application program from virtual fetch and control is returned to the IMS program controller (DFSPCC20) after termination of the application program. Therefore, subroutines and subprograms cannot be managed by virtual fetch.

Unlike preload, virtual fetch always causes a fresh copy of the application program to be invoked. This means that the programs managed by virtual fetch do not have to be link-edited as reusable.

If the application program to be loaded is not supported by virtual fetch, or if virtual fetch has a problem loading the program, IMS uses its current program management interface to load the program. Thus if virtual fetch fails, IMS continues to process the other transactions that can be scheduled in the region as long as the STEPLIB library specification of the data sets that contain the programs is in the dependent region JCL.

10.8.4.3 Changing Programs Managed by Virtual Fetch
To change a program that virtual fetch manages, recompile and relink it into the program library specified in both the virtual fetch address space and message region. While this is being done, IMS continues to invoke the old copy of the program residing in the VIO data set.

After the program is successfully relinked, invoke the virtual fetch refresh routine. This causes the virtual fetch address space to build a new area in the VIO data set, containing the updated copies of the programs. A new hash table is also built. During the build process the IMS dependent regions continue to execute, using the old copies of the programs that virtual fetch manages.

When the new area in the VIO data set is built, virtual fetch activates the new hash table and frees the old area in the VIO data set and hash table.

All of this activity is processed without stopping the dependent region; therefore, all of the programs that can execute in that region can still execute because the region does not have to be stopped.

10.8.5 Comparison of Program Loading Options
The four ways to load application programs are standard program fetch (LINK), virtual fetch (VIO), LLA/VLF, and preload. Each method has its advantages and disadvantages. Table 15 on page 232 compares some of the characteristics of these four methods.

Program fetch (LINK), virtual fetch, and LLA/VLF load a fresh copy of the program into virtual storage each time it is accessed. With preload, the program is fetched only when the program controller is attached. This is usually only once, but it could be multiple times if programs abend in the dependent region.

Virtual fetch and LLA use real storage above 16 MB to load the program. Standard fetch uses only the storage above 16 MB when the page is stolen and then paged in.

Load modules managed by LLA/VLF and virtual fetch are not reusable or reentrant, but we recommend that preloaded modules be reusable or reentrant.
LLA stages active load modules into its VLF data space, which makes the virtual storage usage more efficient than virtual fetch. Although from a performance point of view LLA/VLF and virtual fetch seem to produce the same results, LLA/VLF has more operational advantages than virtual fetch because of its ability to stage active modules. Thus, using LLA/VLF instead of virtual fetch may produce better performance, especially for those modules that are frequently modified, because you do not have to stop and restart the virtual fetch address space.

Installations that change production programs while the online system is up must stop the dependent regions in order to access the new copy of the program. This is true for programs that are managed by standard fetch and programs that are preloaded. If virtual fetch is used to manage the program, you just need to refresh the virtual fetch address space and the new copy of the program is invoked on the next schedule of the application. The dependent regions do not have to be stopped, and therefore the preloaded and standard fetch programs can continue to execute.

<table>
<thead>
<tr>
<th>Program Fetch</th>
<th>Virtual Fetch</th>
<th>LLA/VLF</th>
<th>Preload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loaded once per schedule.</td>
<td>Block paged in once per schedule.</td>
<td>The block is moved from the VLF cache in memory. It is not necessarily paged in unless it is not in real storage.</td>
<td>Loaded once at initialization.</td>
</tr>
<tr>
<td>Nonreusable is okay. Overlay is okay.</td>
<td>Nonreusable is okay. Overlay not supported.</td>
<td>Nonreusable is okay. Overlay not supported.</td>
<td>Reusable is required.</td>
</tr>
<tr>
<td>Virtual = size of largest program.</td>
<td>Virtual = size of largest program.</td>
<td>Virtual = size of largest program.</td>
<td>Virtual = total size of all programs.</td>
</tr>
<tr>
<td>Uses fetch I/O.</td>
<td>Uses (parallel) block page-in of entire program.</td>
<td>Can fetch from VLF data space cache.</td>
<td>No I/O unless pages stolen, then sequential page faults through program.</td>
</tr>
<tr>
<td>&gt; 16 MB real storage used only after page movement or steal.</td>
<td>&gt; 16 MB real storage used when available; entire program paged in.</td>
<td>&gt; 16 MB real storage used when available; entire program paged in.</td>
<td>&gt; 16 MB used after page movement or steal.</td>
</tr>
<tr>
<td>Long path length to load program.</td>
<td>Shorter path length to load program.</td>
<td>Shorter path length to load program.</td>
<td>No path length to load program.</td>
</tr>
<tr>
<td>New copy of program is invoked after MPR is stopped (dynamic BLDL) and restarted.</td>
<td>New copy of program is invoked after virtual fetch is refreshed. MPRs still executing.</td>
<td>New copy of program is invoked after LLA is refreshed. MPRs still executing.</td>
<td>New copy of program is invoked after MPR is stopped and restarted.</td>
</tr>
</tbody>
</table>
Performance measurements conducted on an ES/3090-330S produced the results shown in Table 16.

<table>
<thead>
<tr>
<th>Program Size</th>
<th>Uncached PGMLIB (seconds)</th>
<th>Cached PGMLIB (seconds)</th>
<th>Virtual Fetch (seconds)</th>
<th>LLA/VLF (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 MB</td>
<td>2.660</td>
<td>1.186</td>
<td>0.057</td>
<td>0.065</td>
</tr>
<tr>
<td>1 MB</td>
<td>1.391</td>
<td>0.767</td>
<td>0.048</td>
<td>0.043</td>
</tr>
<tr>
<td>100 KB</td>
<td>0.179</td>
<td>0.091</td>
<td>0.021</td>
<td>0.018</td>
</tr>
<tr>
<td>66 KB</td>
<td>0.076</td>
<td>0.068</td>
<td>0.017</td>
<td>0.017</td>
</tr>
</tbody>
</table>

*Note:* Prefetch time is order of 0.002 second.

### 10.9 IMS Application Program Execution

This section considers the factors that influence the performance of the application program once it has been loaded.

#### 10.9.1 Paging-in the Dependent Regions

If page faulting must occur in the IMS system, confine it, if possible, to the dependent region’s application programs. Three page faults per second in a dependent region can cause approximately a 15% to 20% increase in program elapsed time, or more, depending on the total paging rate.

Real storage considerations for IMS dependent regions include the number of regions, program loading, preload, sharing of common modules, and program structure. If real storage were an unlimited resource, it would be possible to make all IMS application programs resident. However, because real storage is not unlimited, trade-offs for the use of real storage must be made between loading application programs from program libraries and real storage paging (preload). Schedule-end-to-first-DL/I-call time (from the IMS monitor) is used to measure the cost of application program loading and initialization.

Once the program is ready for execution (loaded in virtual storage), page faulting in the application code increases program elapsed time by the duration of the paging activity. If the IMS message regions are paging more than three pages per second, they need additional real storage. Because it is not possible to page fix dependent-region storage, the contention for real storage must be reduced. Normally this contention is caused by running too many lower-priority batch initiators in the system. Each swapped-in initiator requires at least 20 KB of fixed LSQA space, plus the working set of the batch job. Another cause is running batch jobs with very large working sets. Sorts, for example, can use all of the storage available in the address space. Therefore, specify their REGION= size on the EXEC statement or the MAXSIZE parameter as a low value, such as 320 KB. Most installations allow the size to default to MAX, which uses all the storage available in the region of the sort job.

If your system is a fairly well tuned and response times are still unacceptable because of paging in the message regions, additional real storage is required.
Possible responses are to trim the storage requirements of the IMS control region, the DL/I address space, the CSA, and the operating system.

A way to reduce the page faulting in the IMS regions is to implement the storage-isolation function of MVS. Care must be taken not to overcommit this function, as system performance could be degraded to an unacceptable level. If used correctly, this function can maintain response at a consistent and acceptable level in a high paging system. This is true as long as pages can be stolen from batch or TSO users.

### 10.9.2 Dependent Region Dispatching Priorities

This is discussed in detail in 6.3, “IMS Dependent Regions” on page 119, but it is summarized in this section for completeness.

Assign the dispatching priority of the dependent regions above batch and TSO users but below the IMS control region and other IMS address spaces. Consider BMPs as dependent regions, and assign them a dispatching priority above batch.

Give the message regions a higher dispatching priority than the BMP regions. The priority is unique for each message region if IMS class scheduling is being implemented.

If class scheduling is not being implemented, all the message regions execute the same transaction classes, and the WLM is not operating in goal mode, then specifying a rotate priority is adequate to maintain consistent response times. Mean-time-to-wait dispatching priorities can also be used, but normally this has no dramatic effect on response time unless transactions that issue many I/Os per call are being processed.

### 10.9.3 COBOL Options

The various COBOL compile options can have a very significant effect on the CPU consumption of an IMS system. The wrong choice of option can easily account for 15% additional IMS CPU usage.

COBOL II is used as the example.

#### 10.9.3.1 Compiler Options

The recommended options are RENT, RES and NODYNAM.

RENT prevents COBOL subroutines from being linked with the main program. Instead, they are dynamically called at execution time. This demands that the COBOL subroutines be preloaded or made resident once loaded (see LIBKEEP below).

NODYNAM causes prenamed user-written subroutines to be linked with the main program (CALL 'literal' rather than CALL variable_subroutine_name). This gives the best performance, but at the cost of larger load modules. However, in cases where many user-written subroutines exist, or some subroutines are very large, the option DYNAM may be better. This option may also be chosen to simplify updating of subroutines without the need to relink all the application programs. In this case, preload the user subroutines in each dependent region. If this is impractical, use LLA/VLF to manage the subroutines.
10.9.3.2 Runtime Options
Always specify LIBKEEP. It causes any COBOL subroutines that get loaded to remain in memory until the dependent region is terminated. If no COBOL subroutines are preloaded, then LIBKEEP must be used to avoid serious performance degradation. Used with preload of COBOL subroutines, it does no harm and catches any subroutines that were inadvertently omitted from the preload list. IMS preload has the advantage that reentrant subroutines can be preloaded in the PLPA, so that only one copy is required for the whole system. See 10.8.2, “Preload” on page 224 for further details.

10.9.3.3 Calling COBOL from Assembler
Some customers use an assembler module as the “Main Program” loaded by the dependent region. This routine then calls one or more COBOL routines, depending on the input message content. In this case, every time a COBOL routine is called, the COBOL environment in the region is reinitialized, causing significant CPU overhead. It is possible to perform the COBOL initialization only once by having the main assembler program call IGZERRE before anything else. This makes it seem as though the assembler program is a COBOL program.

10.9.4 Common Execution Environment (LE/370)
The LIBKEEP option is not available in an LE/370 environment. However, an equivalent feature, LE/370 library routine retention, is recommended for use with message processing programs written in COBOL, PL/I, or C that use LE/370 in an IMS/TM dependent region.

When running programs in an IMS/TM message-processing region that require LE/370, you can get an improvement in performance if you use the LE/370 library routine retention along with the existing PREINIT feature of IMS/TM. When library routine retention is used, LE/370 keeps certain LE/370 resources in memory when an application program ends. In this way subsequent invocations of programs that use LE/370 are much faster, since the LE/370 resources left in memory are reused.

When library routine retention is used, the resources that LE/370 keeps in memory include the following:

- LE/370 runtime load modules
- LE/370 storage associated with the management of the runtime load modules
- LE/370 storage for startup control blocks

To use LE/370 library routine retention in an IMS dependent region, you must do the following:

1. In your JCL or procedure used to bring up IMS dependent regions, specify that you want IMS to invoke dependent-region preinitialization routines. This is done by specifying a suffix on the PREINIT keyword of the IMS dependent-region procedure.

16 LE/370 library routine retention is available with APAR PN63666. See information APAR II08342 for the LE/370 installation and customization guide changes for this APAR, and see information APAR II08343 for the LE/370 programming guide changes for this PTF. APAR PN63674 includes the changes required for the COBOL component of LE/370 for this function. No changes were required for the PL/I component of LE/370 or the C component of LE/370 for this function.
2. In the DFSINTxx member of IMS.PROCLIB (where xx is a suffix specified by the PREINIT keyword), include the name CEELRRIN.

When the module CEELRRIN is invoked by IMS, LE/370 library routine retention is initialized. There is no requirement to relink with LE/370 to use this new function.

10.9.5 Program Elapsed Time

Assuming that paging is not an issue and that the region is not short of CPU resource, possible reasons for long execution times include these:

- A large number of DL/I calls
- An excessive number of I/Os per DL/I call
- Poor database I/O performance
- Excessive PI WAITs (contention with other programs)
- Excessive deadlocks
- Subroutines not preloaded (including compiler subroutines (See 10.8.2, "Preload" on page 224).
- Excessive time in DB2 (This problem is clearly shown by IMS Monitor. See Chapter 15, “Performance of IMS with DB2” on page 329 for further information.)
- Use of PARDLI=1 in a BMP. This serializes all DL/I calls through the DL/I separate address space. The option is intended for use in development and test systems to reduce the chance of U113 abends. Production BMPs should use PARDLI=0 (the default).

Some of these issues may suggest that you need to review database health (free space and state of disorganization), database design, or application design.

10.9.6 Detailed Monitoring of Program Execution

The IMS call summary report (see 4.2.6, “DL/I Call Summary Report” on page 102) provides a very good view of the average activity of each program. If you need an even more detailed view of individual executions of selected programs, then we recommend that you run the IMS monitor and use IMSASAP II to produce a program trace for the selected transactions.

Program trace is available in two forms, short or long. The difference is that the long form also reports the individual database IWAITs for each DL/I call.
Chapter 11. IMS Database Access Performance

This chapter discusses IMS system logic and parameters affecting IMS database performance. Discussion of the many factors involved in IMS database design for optimum performance are to be found in specialized documentation of database design issues and are not within the scope of this chapter.

11.1 Introduction to IMS Database Buffers

IMS database buffers are the primary interface between the databases and IMS processing regions. Many performance issues affecting IMS databases relate to the passage of data through the database buffers and can be solved by using the various buffer reports available from the IMS monitor and the output of the /DISPLAY POOL command on an IMS terminal.

IMS uses different types of database buffers, according to the type of database and the processing options used.

They can be divided into three distinct categories:

- OSAM buffers
- VSAM buffers
- Fast Path buffers

11.2 OSAM Databases

The OSAM access method is unique to, and is supplied with, IMS. IMS communicates with OSAM using the OPEN, CLOSE, READ, and WRITE macros.

An OSAM database has these characteristics:

- An OSAM data set can be read using either the BSAM or QSAM access method.
- An OSAM data set does not need to be formatted before use.
- An OSAM data set can use fixed-length records blocked or unblocked.
- An OSAM data set can span multiple volumes.
- An OSAM data set has a 4 GB size limit.

Detailed information on how data is organized in an OSAM database can be found in the IMS/ESA V5 Administration Guide: DB (SC26-8012) For information about defining OSAM subpools, refer to IMS/ESA V5 Installation Guide Volume 2: System Definition and Tailoring (SC26-8024).

11.2.1 OSAM Buffer Pool

There is one OSAM buffer pool, typically divided into multiple subpools. A subpool is a set of buffers all of the same size, and the OSAM buffer pool is allowed to have multiple subpools that all have same-size buffers. The definitions of the subpools are put in the DFSVSMxx member (DFSVSAMP DD statement for batch).
Included with the pool definitions is the allocation of databases to subpools. This is useful, for example, for isolating less-used databases from the effects of much busier and “buffer-greedy” databases.

11.2.2 OSAM Sequential Buffering

The normal OSAM buffering read technique accesses one block with each I/O operation, a single block read. Sequential buffering (SB) reduces the time needed for I/O read operation in three ways:

- By reading ten consecutive blocks with a single I/O operation.
- By monitoring the database I/O reference pattern and deciding whether it is more efficient to satisfy a particular I/O request with a sequential read or a single-block read. This decision is made for each I/O request processed by SB.
- By doing look-ahead buffering (asynchronous read-ahead) and so achieving read I/O operations overlapped with processing and with other I/O operations for the same application (batch and BMP regions only).

In general, any IMS application program or utility that sequentially processes OSAM database data sets may run faster using sequential buffering. The time that is saved is difficult to predict, because so many factors affect the elapsed time of a job.

Some of the programs and utilities that may benefit from the use of sequential buffering are:

- IMS batch programs that sequentially process databases
- BMPs that sequentially process databases
- IMS utilities, including:
  - HD reorganization unload
  - Online image copy
  - Database scan
  - Database prefix update

11.2.2.1 Benefits of Using Sequential Buffering

By using sequential buffering for programs and utilities that sequentially process OSAM databases, users can:

- Run existing sequential application programs within decreasing batch window times. For example, if the time set aside to run batch application programs is reduced by one hour, users may still be able to run all programs that they normally run within the reduced time period.
- Run additional sequential application programs within the same time period.
- Run some sequential application programs more often.
- Make online image copies much faster.
- Reduce the time needed to reorganize databases.
11.2.2.2 Specifying Sequential Buffering

The sequential buffering control statements are in a data set with the DDname of //DFSCTL. You can provide a //DFSCTL DD statement in the JCL of batch jobs and IMS dependent regions.

The sequential buffering parameters (SBPARM) control statement is used to:

- Specify which I/O operations to buffer by sequential buffering
- Override SB default parameters

**Note**

When multiple SBPARM control statements are specified, applying to the same I/O operation, the \( n \)th control statement overrides the \((n-1)\)th control statement. This allows you to specify defaults on the first statement that can be overridden by subsequent statements.

The SBPARM control statement has two types of keyword parameters:

- Option parameters, such as ACTIV and BUFSETS, which set or override an SB option or parameter
- Qualification parameters, such as DB, PCB, DD, and PSB, which specify the I/O operations to which the SBPARM statement applies

Figure 76 shows SBPARM control statements.

```
SBPARM ACTIVE=COND, DB=SKILLDB, BUFSETS=6
SBPARM ACTIVE=COND, DB=SKILLDB, BUFSETS=6
SBPARM ACTIV=COND, DB=SKILLDB, PCB=LABEL1 / * multiple PCBs with same DBD
```

**Figure 76. Sequential Buffer Control Statements**

We recommend that users specify ACTIV=COND for batch and BMP jobs that sometimes or always process a database sequentially. SB is not recommended for short-running MPP or Fast Path programs because of the overhead associated with initializing SB for each program execution.

Sequential buffering can also be selected through specification in the PCB. For a detailed explanation of SBPARM parameters, see the *IMS 5.1 Installation Guide Volume 2: System Definition and Tailoring (SC26-8024)*

11.3 VSAM Databases

A key factor influencing the number of database I/Os per DL/I call is the actual design of the database. Selection of indexed fields, the use of pointers, the placement of segments within a structure, and choice of database organization are but a few of the critical design criteria that affect this value. Database design changes, however, have a strong impact on the system and need to be planned. Consider them a long-term measure. Initially evaluate the size and efficiency of the
database buffer pools to ensure that they are not the cause of a high rate of database I/Os and, hence, performance problems.

11.3.1 VSAM Buffer Pool

VSAM local shared resource (LSR) pool contains buffers used by the VSAM index and data components. Buffers of equal length are combined into subpools. The subpools are allocated by CI size and contain a minimum of three buffers.

11.3.1.1 VSAM Buffer Pool Size

Buffer sizes must be a power of 2 and a minimum of 512 bytes. The size and structure of the VSAM pools are defined by control statements in the DFSVSMxx member of IMSESA.PROCLIB (pointed to by the VSPEC= parameter on the IMS EXEC statement). If VSAM databases are not used, do not define VSAM subpools.

If used, define the VSAM subpools in such a way that each control interval size used by the IMS databases resides in its own subpool. In other words, specify a buffer pool definition statement for every CI size in the DFSVSMxx member of IMSESA.PROCLIB.

The minimum acceptable VSAM database subpool size can be estimated by adding the average number of buffers required to hold the database records for transactions that can concurrently execute in all the dependent regions, including BMPs.

11.3.1.2 VSAM Buffer Pool Use

You can define up to 16 LSR pools with up to 11 subpools each. This capability provides the opportunity to direct data and index CIs to different subpools. Having separate subpools eliminates contention for data and index buffers. To achieve optimum performance in each subpool, plan data usage in such a way as to eliminate contention of buffer usage.

The target hit ratio for an index subpool is around 90% and for a data subpool, around 60%. To calculate this hit ratio, refer to the VSAM buffer pool report of the IMS Monitor. Use the formula in Figure 77 to calculate the ratio.

\[
\frac{F}{(F + R)} = \text{HIT RATIO}
\]

where

\[F = \text{Number of times CONT INT requested already in pool}\]

\[R = \text{Number of INT read from external storage}\]

Figure 77. VSAM Hit Ratio Formula

From the values in Figure 86 on page 265, the hit ratio is calculated to be:

\[
\frac{33372}{33372 + 9926} = 77.1 \text{ Percent}
\]
For this example, assume that the buffer subpool is the DATA subpool and therefore the hit ratio falls well within the guidelines.

### 11.3.1.3 Other VSAM Buffer Pool Considerations

The indicator of whether minimum acceptable subpool size achieved is the IMS monitor VSAM buffer pool report. If this shows that forced writes are taking place to make space for application reads, then too few buffers are defined for all the concurrent applications.

A ratio of 0.5 I/Os per DL/I call has been found to be acceptable from the observation of many IMS systems. A higher number of I/Os per call indicates that an investigation into tuning the database buffer pool, the databases, or the application programs should be undertaken. If the database block size and buffer pool parameters have been adjusted and the I/O rate is still high, review Chapter 12, “IMS Application and Database Design for Performance” on page 279. The ratio of 0.5 I/Os per call is a general guideline. A higher I/O rate may be acceptable if it is necessary for the functioning of the application, but do not accept it without investigation.

Increasing the number of buffers in a subpool may not decrease the number of I/Os to that subpool when a BMP or online image copy is reading a database sequentially and causing many I/Os. If it is possible, reschedule the BMP to execute during an off-peak period.

The major tuning considerations for the VSAM database buffer pools are:

- Review VSAM buffer reports from the IMS monitor to determine whether I/Os are being forced to make space for reads.
- Fix the VSAM buffer control blocks.

If BMPs are executing during the IMS monitor run, they could affect the number of I/Os in the buffer pools. Increasing the size of the buffer subpools may not change the number of I/Os because of the way the BMPs may be referencing the different databases. Use the IMS monitor without the BMPs running to determine the number of I/Os in the buffer pools.

### 11.3.2 VSAM Hiperspace Buffers

Hiperspace* is an MVS/ESA facility for storing data, specifically geared for high-performance reads and writes, in expanded storage. IMS provides support for VSAM Hiperspace buffering capability.

Hiperspace buffering offers potential DL/I performance improvements through VSAM I/O avoidance. This is achieved by reducing the requirement for DASD I/O as VSAM finds reassessed buffers in Hiperspace.

The following benefits can be obtained by using VSAM Hiperspace:

- Significant improvements in VSAM buffer hit ratio
- VSAM I/O reduction
- Reduction in internal elapsed time and region occupancy

By reducing the number of DASD I/O operations, you can reduce IMS internal transaction processing times. Although using Hiperspace does not by itself save
CSA, it is possible to obtain CSA savings if IMS transaction processing time reduction allows you to reduce the number of IMS message regions.

When considering VSAM Hiperspace implementation, weigh a number of factors to assess potential benefits. Evaluate each of the following as it applies to your installation:

- A primary consideration is to determine the amount of your overall database I/O activity that is VSAM. OSAM databases are not eligible for Hiperspace, so only the percentage of your total read I/O that is VSAM can benefit.

  Remember to factor in the 4 KB increment consideration. Hiperspace operates on 4 KB blocks; control interval sizes under 4 KB either do not participate or must move up to the 4 KB subpool.

- The total amount of VSAM data is important for two reasons: first, to determine how much data is competing in each VSAM DL/I pool; and second, to determine how many Hiperspace buffers and how much expanded storage are needed for that data.

- Reaccess patterns of the VSAM data are a very significant consideration. Even with large amounts of VSAM data, if a relatively small subset of the data tends to get accessed frequently, Hiperspace buffering is well suited. On the other hand, if your data access patterns are mostly random, spread across much of the data, Hiperspace may be of little benefit.

- Hiperspace benefits read access only; it offers no benefit to database updates. Hiperspace buffering benefits are more visible for data that has a high read-to-write ratio.

- Adequate available expanded storage is important. If you plan to specify a large number of Hiperspace buffers, be sure you have enough expanded storage to contain them. The VSAM Hiperspace implementation uses what is referred to in some literature as expanded storage only (ESO) or cache-type Hiperspace, to differentiate it from standard or scroll-type Hiperspace.

  Standard Hiperspace is available to all programs, both authorized and unauthorized, and can be backed by expanded storage or, if necessary, auxiliary storage. ESO Hiperspace is never migrated to auxiliary storage. ESO Hiperspace does not have to be backed because, if the system becomes constrained and starts stealing expanded storage frames from Hiperspace (possibly causing a read or write to fail), then IMS can always revert to using DASD to retrieve the lost data. This concept leads to an important storage consideration.

  Because ESO Hiperspace has a fallback mechanism, the requirement to provide sufficient expanded storage to meet the needs of temporary spikes in other system activity is alleviated. MVS automatically takes the expanded storage frames it needs for any higher-priority work. More typically, however, IMS is the priority application, and it may not be desirable to allow the system to steal Hiperspace frames. In this scenario, storage isolation can be used to request that the system not steal Hiperspace storage from IMS.

  The system resource manager (SRM) storage isolation function allows an installation to override the normal SRM least-recently-used algorithms in determining the amount of processor storage to be allocated to an address space. The storage isolation function can be used to set a minimum amount of central and expanded storage that an address space requires. It may also be used to indicate acceptable paging rates for an address space. The
Hiperspace pages created by an address space (DLISAS) are considered part of the address space working set for purposes of storage isolation. This consideration becomes particularly important if you plan to specify a large number of Hiperspace buffers.

To isolate storage for Hiperspace, add a unique performance group to your IEAIPSxx, SYS1.PARMLIB member, and code that group in your DLISAS JCL (PERFORM=x).procedure. The PWSS parameter is specified with minimum and maximum storage values.

- I/O subsystem load is also important. If your system is I/O constrained and VSAM database activity is involved, Hiperspace buffering may be particularly useful. This constraint might be a channel, a control unit, or an individual DASD volume. The I/O reductions afforded by Hiperspace could be enough to eliminate the constraint. This has the potential for a dramatic throughput improvement.

- VSAM database candidates for Hiperspace usage include these:
  - Small, heavily referenced databases. Less Hiperspace (expanded storage) is required to cache a larger percentage of a small database.
  - Databases with skewed access patterns. A database that has a small percentage of the total amount of data but is frequently reaccessed is a good candidate.
  - High read:write ratio databases. Hiperspace benefits read access; therefore, databases that have less update activity have a greater potential for I/O reduction.
  - Large index data sets. Index data sets are typically very good candidates because they are relatively small and are frequently reaccessed. If an index data set is small enough, it is more feasible to contain most of it within the DL/I VSAM pool. A larger index data set may need too many DL/I buffers. Adding Hiperspace buffers is a good alternative.

Refer to Figure 78 on page 244 for buffer pool allocation that includes an example of Hiperspace allocation.

### 11.3.3 VSAM Background Write

This is a VSAM-only feature that uses a lower-priority asynchronous task to write out updated buffers whenever a subpool becomes completely full of updated buffers. The objective is to make buffers eligible for stealing so that applications do not have to wait on a space write before being able to read in a new CI. It is requested by a parameter in the DFSVSMxx member (DFSVSAM DD statement for batch), BGWRT=(YES,n). The value n is the percentage of buffers examined, as explained below.

Each time the buffer manager obtains space in a subpool, it acquires the buffer least recently used. It also examines the next higher buffer on the LRU chain. If the contents of that buffer are modified, a return code is passed in the RPL to IMS. This return code tells IMS buffering services to activate (post) the BGWRT PST, which is then dispatched through normal IMS scheduling. This BGWRT task is dispatched as an asynchronous low-priority task in either the control region (with LSO=Y) or the DL/I separate address space (with LSO=S). Consequently, the I/Os cannot be charged to a specific dependent region.
OPTIONS,BGWRT=(YES,20)
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON,DLOG=ON,SUBS=ON
POOLID=DFLT,FIXBLOCK=YES,FIXINDEX=YES
VSRBF=512,800
VSRBF=1024,20
VSRBF=4096,100
VSRBF=8192,32,I
VSRBF=8192,120,D
VSRBF=12288,10,HS500,HSR
POOLID=CUST,FIXBLOCK=YES,FIXDATA=NO
VSRBF=4096,1000
POOLID=SHIP,FIXBLOCK=YES
VSRBF=4096,20
DBD=CUSTINDEX(1,DEF1)
DBD=CUSTDATA(1,CUST)
DBD=ORDERS(1,DEF1)
DBD=PRODUCTS(1,DEF1)
DBD=BILLING(1,DEF1)
DBD=SHIPPING(1,SHIP)
IOBF=(2/zerodot48,12,N,Y)
IOBF=(4/zerodot96,25,N,Y)
IOBF=(6144,5/zerodot,N,Y)
IOBF=(1/zerodot24/zerodot,5,N,N)
IOBF=(18432,5,N,N)
SBONLINE,MAXSB=1000
OLDSDEF OLDS=(/zerodot/zerodot,/zerodot1,/zerodot2,/zerodot3,/zerodot4,/zerodot5),BUFNO=/zerodot/zerodot5,MODE=DUAL
WADSDEF WADS=(/zerodot,1,2)

Notes: The parameters that define the Hiperspace requirements are:
VSRBF=buffersize,number-of-buffers,type(optional),HSn.HSR

HSn: n denotes the number of Hiperspace buffers.
HS500 means 500 Hiperspace buffers.
HSR: denotes the expanded storage is Required for IMS to start
HSO: would denote that expanded storage is Optional

Figure 78. Example of DFSVSMxx Parameters Using Hiperspace

The BGWRT task processes each VSAM subpool in turn. For each subpool, it
examines the specified percentage of buffers on the LRU chain. Any modified
buffers it finds are written to DASD. The objective is to avoid a situation where the
subpool is filled with updated buffers, because, if this happens, every read request
will first require a write to make a buffer available. Background write does not
prevent reuse of these buffers. If a subsequent request requires the data in the
buffer before the buffer manager needs the buffer for a new block, the buffer is
used to satisfy the request and is placed on the top of the buffer-use chain.

As mentioned, if the BGWRT option is turned on, the IMS monitor reports may not
charge the I/Os for the VSAM buffer pool to the program that updated the record.
The I/O count is reflected in the buffer pool statistics only. This option was
designed to improve batch DL/I performance, but it can help an online system as
well if BMPs or long-running MPPs are updating many database records.
For most IMS message-processing systems, the buffer pools are easily big enough to hold all the updates from all the concurrent transactions. Thus, background write normally never gets scheduled, and the default values (YES,34) are fine.

11.4 IMS Fast Path Databases

IMS Fast Path includes two database organizations: the data entry database (DEDB) and the main storage database (MSDB).

DEDBs are similar to HDAM but have some significant differences that provide even higher performance, capacity, and availability. These differences demand a rather different approach to tuning.

11.4.1 Main Storage Database

The MSDB provided a valuable set of facilities in the days when computers were memory constrained. In recent years, however, the amount of memory available on systems has increased dramatically. The virtual storage option (VSO) for DEDBs provides a far richer set of facilities than is available with MSDBs.

The following paragraphs describe MSDBs, but are essentially for historical purposes only.

The MSDB is a single-segment-type database that resides in main memory. It provides a high degree of parallelism and performance and, as such, is ideally suited to data that is heavily updated. To provide fast access and allow frequent updates to the data, MSDBs reside in ECSA and can be optionally page fixed to give even higher performance. Deciding when to use an MSDB, and which ones to page fix, involves a trade-off between the required application performance and the amount of real storage available.

Although all updates to MSDBs are logged during sync point processing, the data is written to the in-memory MSDB only. The MSDBs themselves are written out to DASD when one of the following occurs:

- Simple checkpoint
- Shutdown
- Restart

The use of MSDBs is an efficient way of minimizing I/O, so few performance problems are experienced. Where problems do occur, it is often because MSDBs tend to be small and thus are more likely to cause contention in a busy system. To minimize resource contention on MSDBs, there are several points to consider for application design. For details, see the IMS/ESA V5 Administration Guide: Database (SC26-8012)

11.4.2 Data Entry Database

A DEDB is a hierarchical database designed to provide efficient storage and access to large volumes of data with a high level of availability.
11.4.2.1 Performance Problems
As is the case with DL/I databases, performance problems do occur and can be solved as follows:

- DASD or channel contention for I/O on DEDB
  
  One possible solution to the problem of DASD contention with DEDBs is the use of area data sets. It may be possible to identify the parts of the database that are heavily accessed (called *hot spots*), and move these records either to another existing area or to a new area. Create the new area on a different DASD and controller if the problems are related to hardware performance.

  Two other potential solutions also exist:
  
  - For data sets that use noncached controllers, multiple area data sets (MADS) may help. IMS uses a round-robin algorithm when reading from MADS, and by placing the area data sets onto different DASD volumes, you can spread the read I/Os across several volumes.
  
  - Otherwise, use cached controllers for your most heavily accessed databases.

- OTHREAD contention
  
  When buffers waiting to be written start queuing for OTHREADs, buffer contention increases because the locks on the data in the buffers are not released until the buffers are written to DASD.

  Split the area and spread it across several DASD volumes.

- Increased I/O or CI contention for IOVF or DOVF
  
  Reorganize the database.

- Overflow buffer allocation (OBA) latch wait
  
  Increase the normal buffer allocation (NBA).

RMF and the Fast-Path log analysis utility can monitor the above items. The IMS DBTools pointer checker utility can be used to monitor the usage of IOVF and DOVF within your databases. Do this periodically to ensure that the performance of these databases is not being impacted by the overuse of OBA.

11.4.2.2 DEDB Using the Virtual Storage Option
In IMS 5.1, VSO provides the IMS user with the best features of both data entry databases and the main storage databases. The definition, access, and operations on a DEDB using VSO function in exactly the same way as for an ordinary DEDB, but the data is kept in virtual storage while the database is open. Thus, programs do not wait for I/Os when segments are accessed.

When a Fast Path program requests a segment from IMS, IMS determines which CI needs to be read from the database. A buffer is allocated from the Fast Path buffer pool, and the CI containing the target segment is read into the buffer. If VSO has been set for this database, IMS first checks whether this CI has already been read into data space. If so, the CI is copied from the data space into the buffer, and the program continues. When the program updates this segment, IMS logs all the segment changes.

Any data entry database can be moved to virtual storage by specifying the VSO keyword on the DBRC commands INIT.DBDS or CHANGE.DBDS. IMS allocates a
data space, and as each CI is read from the database, it is placed in this data space.

Additionally, an area can be defined to DBRC with the PRELOAD attribute, which means it is read into the data space at IMS startup.

Using VSO for a database requires slightly more CPU than when using MSDB. However, VSO is preferred over MSDB for a number of reasons:

- Operational characteristics
  Operationally, MSDBs are problematic. A shutdown of IMS is required in order to add segments. The process of updating the MSDBINIT data set and getting IMS to reload from this data set is manual. A DEDB with VSO is still a DEDB, can be manipulated without impacting the running IMS system, and can be managed by DBRC, allowing for standard image copy and recovery procedures.

- Migration
  Application programs written to access MSDBs can access the equivalent DEDB without alteration.

11.4.3 DEDB Sequential Processing
If you have applications that use a PCB to sequentially process (read or update) one or more areas of a DEDB, consider using HSSP. This is requested by using PROCOPT=Hx. There are no special hardware requirements (unlike the first implementation of HSSP, which was written to exploit sequential caching mode in DASD controllers).

Sequential processing means physical sequential processing, typically driven by unqualified GN calls on the root segments. However, with care, it can also be the result of qualified GN or GU root calls. The essential aspect of the processing is that many or all CIs in the base section of each unit of work are accessed.

When using HSSP, IMS allocates private buffers to the application and uses both chained reads and look-ahead buffering. The update writes are done by standard OTHREADs and hence are asynchronous and use chained write I/Os.

A standard option with HSSP is to take an image copy (called asynchronous image copy) while the HSSP application is running. The current implementation is completely different from the first release. All the operational complexity has been removed, and the performance has been significantly enhanced by (as the name suggests) overlapping the image copying with the application processing.

HSSP provides superb sequential performance by exploiting significant amounts of page-fixed storage for the private buffers.

For a full explanation of HSSP, see the IMS/ESA Version 5.1 Guide (GG24-4302).
11.4.4 IMS Fast Path Buffers

The following IMS control region and dependent region EXEC parameters related to Fast Path buffers are important for performance tuning:

- Control region EXEC parameters
  - DBBF  The total number of buffers
  - DBFX  The system buffer allocation

- Dependent region EXEC parameters
  - NBA  Normal buffer allocation
  - OBA  Overflow buffer allocation

The number of Fast Path database buffers required is calculated using the formula shown in Figure 79:

\[
DBBF \geq (\text{Number of open areas that have SDEP segments})
+ (\text{Sum of NBA for all active FP programs})
+ (\text{Largest OBA allocation for any of the active FP programs, including any specified by CICS for DBCTL})
+ DBFX
+ (\text{Sum of all Fast Path buffers used by CICS (CNBA)})
\]

*Figure 79. Fast Path Database Buffers*

If the number of database buffers requested by DBBF is not large enough, an area open or a region initialization will fail.

11.4.4.1 Normal Buffer Allocation

When a dependent region is started with NBA specified in its execution parameters, it causes the NBA number of buffers to be made available in the Fast Path buffer pool. This number of buffers needs to be sufficient to handle the processing of the vast majority of programs running in that region. These buffers are page fixed when the region starts.

All CIs locked at the exclusive level remain locked until the buffer is released. Buffers that have not been updated are released when:

- The NBA limit is reached (and buffer stealing occurs),
- The program reaches sync point.

Updated buffers are released only when the OTHREADs have completed.

The Fast Path log analysis utility (DFSULTA0) shows the number of buffers actually used by each Fast Path transaction.

Try to aim for having at least 99% of transactions acquire all their buffer needs from NBA.

The number reported on the first line of the Fast Path log analysis exception report is the total number of buffers used, irrespective of whether they were NBA only or NBA+OBA. The “Buffers” line in the report gives the detailed breakdown of the number used within NBA and OBA.
The report can provide a breakdown of buffer usage by type. It details the number of NBA, OBA, NRDB (nonrelated buffers for SDEP and MSDB use), number of buffers stolen, number of times the program waited for a buffer to become available, number of buffers written with OTHREADs, as well as the number of buffer sets used by HSSP and the high-speed reorganization utility.

11.4.4.2 Overflow Buffer Allocation
If your program requires more than its NBA, IMS can provide additional buffers. The number allowed is specified by the OBA parameter on the region procedure. However, IMS permits only a single program to access its OBA buffers at any point in time and uses the OBA latch to enforce this. (Generally, OBA is required only by update programs.)

The OBA latch is released when the holding program reaches sync point. If the latch is not available because another program is using its OBA buffers, the region waits until the latch becomes available. At any time, only the largest OBA requested by a region is page fixed in the Fast Path buffer pool. Allocate sufficient NBA for the majority of work units, so that OBA is rarely used.

11.5 IMS Database Buffer Handlers
This section covers the OSAM and VSAM full-function buffer handlers. The selected functions are those that can be related to external analysis and tuning actions to improve performance.

11.5.1 IMS Batch
Figure 80 on page 250 presents an MVS address space. Within this address space is a stand-alone batch application program that issues DL/I calls. The calls invoke the DL/I interface module (not shown) within the batch address space and then pass to the call analyzer.

A key function of the call analyzer is to make sure the call is syntactically correct and suitable for further processing. The call analyzer is the module that passes Ax status codes back to the program when the call is incorrect. For example, when an invalid call function code is detected by the call analyzer, it passes an AD status code to the application program.

After the call analyzer has approved a call, it is then passed to an appropriate action module for processing. In general, these action modules are known as Retrieve, Insert, Delete, and Replace.

It is the action modules that interface with the buffer handlers to request data (for example, segments) and services (such as marking a buffer altered).

Update database activity requires the before and after images of data to be logged. The action modules are responsible for this logging activity. As shown in Figure 80 on page 250, the buffer handler owns and manages the database subpools. It is also responsible for all database reads and writes. It may do some logging, but usually very little.

When the buffer handler receives a request from an action module, it looks aside into the appropriate subpool first to see if the request can be handled from there. If the contents of the subpool can satisfy a request, then a read (and possibly a write)
are avoided. If the action module request cannot be satisfied from the appropriate subpool, then the buffer handler must read the data requested from the database on DASD into a buffer in the subpool. Overall performance is dependent upon the degree to which database reads and writes can be avoided.

In the stand-alone batch environment, a single MVS TCB is responsible for all processing. All activity is thus serialized within the stand-alone batch program.

11.5.1.1 Buffer Handler Contention
When considering database contention in the stand-alone batch environment, contention may occur within the buffer handler for buffer pool resources.

A stand-alone batch can encounter contention (waiting on a resource) in two forms:

- Contention with OSAM sequential buffer read-ahead
  An OSAM SB overlapped read is an attempt by OSAM to read ahead ten blocks while the application program, or DL/I on its behalf, continues to execute. This execution may result in a request for a block of data that is in the process of being read by the overlapped read. If so, OSAM recognizes this and causes the request to wait until the overlapped read completes. This form of contention has both a positive and negative side. It is positive in that some additional processing overlaps to some degree with the overlapped read. It is negative in that processing had to wait for the overlapped read to complete.

- VSAM background write with other VSAM read activity
  Background write (see 11.3.3, “VSAM Background Write” on page 243) attempts to overlap the write of altered buffers from the buffer-handler subpools while the application program, or DL/I on its behalf, continues to process. In general, the net effect of VSAM background write is positive. Any overlap of processing between background write and our program reduces the elapsed time execution of our program.
The application may, however, request a CI it has already updated, and which, by coincidence, is currently being written out by background write. In this case, the read request waits for the write I/O to complete.

### 11.5.1.2 IMS Batch DL/I Buffer Usage

Performance problems can arise if the number of database buffers specified for batch DL/I jobs are insufficient. The numbers of buffers for VSAM and OSAM are specified in the DFSVSAMP DD data set.

Use the DB monitor with a batch DL/I job to monitor the use of buffers and the I/O profile.

### 11.5.2 IMS DB/TM and DBCTL

Figure 81 shows IMS DBCTL structure.

![Diagram of IMS DB/TM and DBCTL structure]

Figure 81. DL/I Processing in IMS DB/TM and DBCTL

An IMS DB/TM or DBCTL system is a parallel environment with up to 999 application programs using cross-memory services to access IMS code, control blocks and buffers. The databases, for example, are shared by all the applications.

### 11.5.2.1 Online Contention

Figure 82 on page 252 shows an expanded DL/I SAS address space. The diagram shows multiple application tasks (each represented by a TCB) accessing DL/I and encountering several situations:

- A task can be accessing data in a block or CI within the buffer pool to satisfy a given call. Multiple tasks can access the same buffer at a time within the buffer

---

17 The batch IMS monitor (DB monitor) is switched on by the execution parameter MON=Y. A trace output DD statement must be included to collect the monitoring data.

18 255 prior to IMS/ESA Version 5.1.
pool. The only restriction is that updates to a given block or CI are serialized by locking.

- A task can wait on a lock held by another task.
- A task may be in the process of actively placing a log record in a log buffer (logical logging). Logging is a serialized function. If another task wants to write a log record, it has to wait.
- A task may have encountered an exclusive latch within the buffer handler itself and must wait for the latch to be released before it can continue.

Figure 82 also shows that DASD contention is often a fact of life for an online system handling multiple dependent regions or multiple threads from CICS. However, DASD contention is not exclusive to an online system. DASD contention in the stand-alone batch world requires only two or more batch jobs to be accessing data sets on the same volume.

The buffer handler’s activities or responsibilities for an online system are the same as those discussed for stand-alone batch. An online system does, however, have a greater potential for contention. Locking and latching of resources must be done to preserve integrity. Preserving integrity within an online system and delivering performance at the same time are two basic reasons for the sophistication and complexity of online database management software:

- Database resources are locked to prevent two or more tasks from updating the same block or CI simultaneously and to allow for dynamic backout of a unit of work if a failure should occur, thus ensuring data integrity.
As stated, only a single task can be allowed to place a log record in the current log buffer at a time. To ensure that this serialization happens, a task must own the log latch before it can place a log record in the current log buffer.

Contention within the buffer handlers is reflected or reported by the IMS Monitor differently, depending on the access method used. For OSAM, buffer handler contention (wait time) is reported as DL/I call NOT-IWAIT time. For VSAM, each buffer handler contention is reported as an IWAIT. These VSAM contention IWAITs cannot be distinguished from VSAM database reads and writes, except perhaps by their overall duration.

11.5.3 Buffer Pool Management

Each buffer in a subpool has a prefix that contains:

- Buffer ID
- Least recently used (LRU) chain pointer
- Most recently used (MRU) chain pointer
- Relative block number
- Waiting PST anchor
- Buffer attributes

11.5.3.1 Use Chains

Each buffer has a unique identifier. All buffers are chained together on the use chain, which is used to help the buffer handler manage the buffers in a subpool. Soon after database activity is started for an IMS subsystem, all buffers in a subpool are occupied by a block or CI. When a subpool is fully occupied, a buffer must be stolen if a new block or CI has to be read.

The LRU and MRU pointers are really the forward and backward pointers of a given buffer on the use chain. Whenever a buffer is referenced, its pointers are changed such that it becomes the MRU buffer in the subpool. As time goes by and a buffer remains unreferenced, it works its way down the use chain until it becomes the LRU buffer. A buffer handler function called buffer steal searches a subpool when a buffer must be stolen, beginning at the LRU end of the use chain.

11.5.3.2 Buffer Attributes

Buffer attributes are kept in the buffer prefixes. These attributes are used to assist in buffer management.

The following is a list of buffer attributes:

- **Owned**
  - Set when position is established in a buffer.
  - Released when the call is completed or moves to another buffer.
  - A dependent region can own only one buffer at a time.
  - Multiple dependent regions can own the same buffer.

The owned attribute protects the contents of a buffer while the caller (action module) is working in the buffer. Protection helps a task to do its work in as short a period of time as possible without interruption. For example, an owned buffer cannot be stolen. If buffer steal wanted to steal an owned buffer, it would have to wait until ownership was released.

- **Busy** (OSAM), **Locked** (VSAM)
The Busy or Locked attribute is comparable to an exclusive lock on a buffer. It is set when a read or write is in progress into or from a buffer.

- **Locked**

“Locked” in this example refers to a long-term buffer condition. If a write error occurs for a block or CI, the buffer handler attempts to get storage outside the buffer pool and move the contents of the buffer in the subpool to that storage. If the buffer handler’s request for storage fails (highly unlikely, but possible), then the buffer with the write error is locked in the subpool until the associated database data set is closed.

For example, if a write error occurred for this block, and no storage was available for the write error buffer (EEQE), then the buffer is locked.

- **Altered** Set when an update call modifies a buffer.

- **New ID Pending** (OSAM only)

Set when a read into the buffer for a new block is issued. It is reset when the read completes and the buffer prefix is updated for the new block.

- **Empty**

This attribute does not occur very often. All buffers are empty when a subsystem begins execution, but shortly thereafter, every buffer contains a block or CI.

A data sharing environment may be the most common environment in which the Empty attribute is set for an executing system. If a data sharing partner updates a block or CI in one of its subpools, all sharing partners are notified of this update. If they have the same block or CI in one of their subpools they invalidate that buffer by marking it Empty. This is known as buffer invalidation.

### 11.5.3.3 Managing Contention

In an online system, multiple tasks access the buffer handler concurrently, and contention for the resources managed by the buffer handlers is likely.

The following lists of functions for OSAM and VSAM are similar but the order of the functions is different. The first function under each buffer handler indicates the most common manner in which each buffer handler manages contention:

**OSAM Buffer Handler Functions**

1. Subpool latch
2. Buffer attributes
3. Compare and swap

**VSAM Buffer Handler Functions**

1. Compare and swap
2. Buffer attributes
3. Latching

The OSAM buffer handler uses the subpool latch to ensure that only one task at a time can update subpool control information (for example, when performing buffer steal to acquire a new buffer for a DL/I call). If another task also wants to perform a function that requires an update of the control information for that same subpool, then it must wait until the subpool latch is released.
OSAM uses compare and swap logic to serialize some update functions, but not to the degree that the VSAM buffer handler uses it. Compare and swap logic is the preferred serialization mechanism used by the VSAM buffer handler. VSAM never latches (locks) an entire subpool. The VSAM buffer handler could use a latch to serialize task requests but seldom does.

Buffer attributes are commonly used by both buffer handlers to manage contention. For example, the buffer steal routine prefers to steal a buffer that is not owned over one that is owned. If an owned buffer is selected to be stolen, the steal routine must wait for ownership to be released before it can steal it. The steal routine, thus, tries to avoid this wait (contention).

### 11.5.3.4 Buffer Handler Requests
Several types of request are made by the OSAM and VSAM buffer handlers. The list for the OSAM buffer handler is shorter. The main reason for this difference is that the VSAM buffer handler must manage two types of data sets; keyed (KSDS), and nonkeyed (ESDS). The OSAM buffer handler, on the other hand, handles only nonkeyed data sets. These are the request types:

- **OSAM**
  - Search pool for block in range.
  - Release ownership of a buffer.
  - Locate a single block.
  - Locate byte.
  - Create new block.
  - Write blocks altered by PST (purge buffer-queue element chain).
  - Mark buffer altered.
  - Locate byte and mark altered.
  - Mark buffer empty.
  - Purge the pool because of application checkpoint.
  - Purge the pool because of an abend.
  - Do a buffer forced write.

- **VSAM**
  - Locate a logical record by block RBA.
  - Locate a logical record by byte RBA.
  - Purge all buffers of the current PST if an I/O error occurs.
  - Search a subpool for an RBA within a certain range.
  - Mark buffer altered.
  - Do a background write.
  - Locate a logical record and mark buffer altered.
  - Purge all buffers at request of abend STAE routine.
  - Purge subpools when checkpoint or sync point occurs.
  - Locate a logical record whose key is equal to or greater than a given key.
- Locate the first logical record of a database.
- Erase a logical record.
- Insert a logical record within a KSDS.
- Add a logical record to an ESDS.
- Get the next logical record in a database.
- Release PST ownership of a buffer.
- Mark all buffers invalid.
- Locate the given logical record for image copy.
- Get the next logical record for image copy.

11.5.3.5 Buffer Pool Purge

Figure 83 shows buffer pool purge processing flow.

![Buffer Pool Purge Diagram](image)

Buffer pool purge is an important function, because it occurs at sync point for all update transactions and drives all (or most) of the full-function database write I/Os.

When buffer pool purge is invoked, the buffer handlers are required to write all altered buffers to the database data sets. Buffer pool purge is invoked for a dependent region whenever a sync point is reached. This happens frequently for a transaction-driven system. Stand-alone or non-message-driven BMP's also invoke buffer pool purge whenever the application program issues a CHKP or SYNC call (may or may not be frequent).

To facilitate the buffer pool purge process for a program, IMS chains together all the buffers altered by a program to form a single chain of altered buffers anchored off the PST control block. When the application program reaches a sync point, finding the required altered buffers to purge is simplified by following the chain.

The key aspect of this applies especially to very large buffer pools and is that the entire buffer pool does not have to be searched. The purge processor simply follows the chain of altered buffers for the dependent region (PST) that is performing sync point processing. The elements linked together on the chain are buffer queue elements (BQEL). A BQEL is allocated that points to each buffer altered by a dependent region unit of work and these BQELs are chained together off an anchor point in the PST (see Figure 83). The use of BQEL chains provides an efficient method of finding all altered buffers for a given unit of work, and the cost is directly proportional to the number of buffers updated.
**Log Write-Ahead (LWA):** Before any altered buffers can be written to the database data sets, the purge process ensures that all database change log records associated with the altered buffers are physically written to the log.

The buffer prefixes contain the log record identifier of the last change made to an altered buffer. A scan of the BQEL chain to be written reveals the highest log record identifier represented on the chain. A check with the IMS logger tells the purge process the last record identifier physically written to the log (batch log, OLDS, or WADS). If the last record identifier physically written to the log is greater than the log record identifier value reflected in the BQEL chain, then all database change log records are physically on the log device and purge does not have to force a write to the log. If not, the purge process forces a write to the log (and waits on it) before any writes are made to the database data sets:

- **OSAM Buffer Purge**
  - The BQEL chain is read and all altered buffers are accumulated.
  - The altered buffers are sorted in RBA sequence within volume, within subpool.
  - If a subpool is fixed, a single channel program is built for each volume. Within each channel program, altered blocks on the same track are chained together. Once the channel programs are built for a subpool, SIOs are issued concurrently, one for each volume to be written to. Then the same process is completed for the next OSAM subpool and so on, until all OSAM subpools have been purged.

  The process is slightly different if a subpool is not fixed. When the subpool is not fixed, the channel programs are limited to no more than 49 buffers per channel program per volume.

- **VSAM Buffer Purge**
  - VSAM purge is started as soon as the SIOs for the OSAM purge channel programs have been issued. The result is that VSAM purge is overlapped with OSAM purge.
  - Altered buffers (CIs) are each written with a single SIO as they are encountered on the BQEL chain.
  - The VSAM WRITE BUFFER command is passed the CI information (data set and RBA) rather than a buffer address. VSAM then uses an efficient hashing technique to locate the buffer containing the CI.

The write of an altered buffer to a database data set does not destroy the contents of the altered buffers in the subpools. The blocks and CIs that were written are still available in the subpools. The only elements in the subpools that change are the prefixes of the buffers that have been written. After the purge process, the prefixes indicate the buffers are unaltered. Subsequent requests for the blocks and CIs are satisfied from the buffers. No I/O from DASD is required.

**Tuning Comments:** These may be of use:

- Large block sizes are best for programs that sequentially update the database.

  The assumption is that the database data set is accessed in physical storage sequence. Therefore, the larger the block or CI size, the fewer the number of blocks and CIs that have to be read and written.

- Maintain database record (DBR) locality of reference:
– Attempt to set things up so that a record spans as few blocks or CIs as possible. If this is done, then a given buffer pool is used most efficiently. In the past, a 4 KB block or CI size became a default standard for the data portion of a database, regardless of how large a record might be. Part of the reason for this 4 KB standard was the lack of virtual and real storage for buffers. Virtual storage is less of a concern as the buffer pools are now above the 16 MB line for both IMS DB/TM and DBCTL. Assuming adequate real storage you can now tailor block and CI sizes to the size of a database record.

– Set SCAN=0 in your HDAM and HIDAM DBDs. This tends to keep a database record located within the most desirable cylinder. When SCAN is greater than zero, the HD space search algorithm is given the opportunity to spread the segments of a database record across multiple cylinders (and blocks or CIs).

– Programs inserting segments need free space in HD databases. The use of free space percentage factor (FSPF) tends to spread a database record at load time across multiple blocks and CIs. The use of free block frequency factor (FBFF), however, gives a better chance of placing multiple inserts in the same block or CI. This also applies to the use of variable length segments that can be split when replaced. Note, however, we do not recommended using FBFF for HDAM (see 12.2.1.2, “Hierarchic Direct Access Method” on page 280).

– Schedule database reorganizations regularly. Nothing brings a database record back together better than a timely reorganization. It is best to reorganize regularly, rather than waiting until the database performance starts to degrade.

• OSAM writes out altered blocks in one of two ways:
  – A block at a time as a result of an altered buffer steal (a forced write or space write).
  – With OSAM chained write as a result of an application sync point.

Because OSAM chained write is far superior in terms of performance when compared with altered buffer steal, balance the size of OSAM subpools with the frequency of application program checkpoints. Using OSAM chained write may eliminate all altered buffer steals.

• Even though it is rarely used by online systems, background write is the most efficient way to write out altered buffers with VSAM. Never turn it off, but be wary of setting the Background Write percentage too high. Background write must steal altered buffers before it can write them to the database data sets. When an altered buffer is stolen, a log write ahead (LWA) operation may be required. LWA is especially likely if the background write percentage is set too high. We recommend that you use the default value (YES,34) in the DFSVSVMxx member for online systems (see 11.3.3, “VSAM Background Write” on page 243).

11.5.3.6 OSAM Buffer Stealing
Buffer stealing is an important topic as it occurs every time a block or CI is read into a subpool.

Stealing Algorithm: OSAM and VSAM do not use the same algorithm to select a buffer to steal. Understanding the two algorithms can help you set up your buffer...
Figure 84. OSAM Buffer Stealing - Algorithm

pools in an improved fashion and minimize the cost of buffer stealing. The cost to be avoided is primarily path length, but it may also include log write-ahead activity.

**Overview:** OSAM buffer steal uses one to three passes in searching for the most appropriate buffer to steal. In the first pass, it looks at a few buffers that are the least recently used in the subpool. If it finds an ideal buffer, the search ends and that buffer is used. Otherwise, it allocates a use level to each buffer examined to reflect its appropriateness for stealing. Having examined this small set, it selects the one that has the lowest use level and, if it is acceptable, that buffer is stolen.

If the first pass is unsuccessful, it tries again, but looks at considerably more buffers in the chain of buffers least recently used. If this also fails to find a usable buffer, OSAM makes a final pass, and this time with less rigid criteria for an acceptable buffer to steal.

**Setting the Use Level:** Figure 84 contains a table that the OSAM buffer steal routine uses when setting the use level of each buffer examined. If a buffer is found with a use level of zero, the search ends, and that buffer is stolen.

The column headings are explained below:

- **OWNED**
  Ownership is a buffer attribute and indicates that a dependent region is actively positioned within the buffer. In order to steal an owned buffer, the steal routine must wait for ownership to be released.

- **PSTLR**
  PSTLR is a mnemonic for PST last referenced. Part of the decision to steal a buffer depends on which PST (dependent region) last referenced the buffer being evaluated by the steal routine as a candidate to be stolen. OSAM prefers to steal a buffer that was last used by the requesting region.
• BUSY

Is the buffer being considered by the steal routine busy writing a block? If it is busy being written and the steal routine does select this buffer to be stolen, the steal routine must wait until the busy attribute is reset, that is, until the write is complete.

• ALTERED

Is the buffer being considered by the steal routine altered? If so, and the steal routine selects this buffer to steal, then the steal routine must wait until the altered block in the buffer is written to the database data set. It may also first have to wait on LWA if the database change records associated with the altered buffer have not yet been physically logged.

• USE LEVEL

Each buffer that is evaluated, or considered, by the steal routine is given a value based on one or more of the attributes just discussed. This value is the use level. A buffer with a low use-level value is more desirable to steal and, for the buffers evaluated, the buffer with the lowest use-level value is the one that will be stolen. As shown in Figure 84 on page 259, an empty buffer or one that is not owned, busy, or altered, and was last referenced by this PST has a use level of 0 and is always stolen before any buffer that has a higher use level.

Use levels of 0 or 1 are cheap steals. The steal routine does not have to wait on anything in order to steal the buffer. However, stealing a buffer with a use level higher than 1 requires the steal routine to wait before the buffer can be stolen. The wait end when

– Ownership is released.
– A busy condition ends.
– An altered buffer is written.
– Or some combination of these conditions occurs.

• The bottom three lines of Figure 84 on page 259 show three unique states that a buffer may be in and the highest use-level values are associated with these states.

A use level of 9 means a block is in the process being read and once the read completes, the ID of the buffer prefix will be changed. Because a read in progress is made only to satisfy a request in progress, the stealing of this buffer would disrupt the request in progress and is, therefore, not allowed.

A use level of 10 (buffer locked as a result of a write error) means OSA is unable to write the contents of the buffer to the database data set on DASD. IMS, for both OSAM and VSAM, does not terminate if a write error to a database data set occurs. Instead, it attempts to get storage outside of the subpool for the block or CI that could not be written and maintains it in storage until IMS terminates. If the request for storage fails, for whatever reason, IMS locks the buffer with the write error in the subpool. IMS always retries write errors when the database in question is closed.

Setting the Number of Buffers to Examine: The steal routine starts looking for a buffer to steal at the LRU end of the use chain. It evaluates a buffer and gives it a value (use level) in its search for a buffer to steal.
An OSAM subpool is searched in up to three passes, and the number of buffers is incremented each time. Figure 85 on page 261 describes how the number of buffers is determined for an increment. Keep in mind that any buffer evaluated with a use level of 0 is immediately stolen and the search ends.

For the first and second increments, note that the steal routine steals buffers with use levels between 0 and 5. The third increment selects buffers with values up to 8.

Referring again to Figure 84 on page 259, notice that, all other things being equal, OSAM will steal a buffer last used by the same region (PSTLR=EQ).

Use levels 0 through 5 are eligible for selection in the first and second passes. That is, the first and second passes can steal a buffer that is busy writing or altered, but not owned.

Because a dependent region can own no more than one buffer at a time in a subpool, if you have more buffers in a subpool than you have dependent regions, the steal routine probably never has to steal an owned buffer.

In Figure 85, the steal routine holds the subpool latch while evaluating buffers to be stolen, which means no other task (dependent region) has access to the subpool except to process its currently owned buffer. If the steal routine must wait on an event (write of an altered buffer, for example) it releases the subpool latch prior to waiting.

In general, the OSAM steal routine is very efficient. Given an adequate number of buffers in a subpool, buffer steal finds a buffer to steal within the first pass almost

<table>
<thead>
<tr>
<th>Increment</th>
<th>Eligible Use Levels</th>
<th>Example Buffer Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>0 to 5</td>
<td>60 buffers, 15 regions</td>
</tr>
<tr>
<td>2nd</td>
<td>0 to 5</td>
<td>30</td>
</tr>
<tr>
<td>3rd</td>
<td>0 to 8</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 85. OSAM Buffer Stealing - Number of Buffers to Examine
every time. If altered buffer steals occur (reported by the IMS monitor in the
database buffer pool reports; see 11.6.2, “IMS Monitor OSAM Buffer Pool Report”
on page 269), consider increasing the size of a subpool or increasing BMP
checkpoint frequency. Do not allow transaction-driven systems to incur altered
buffer steals. If they do occur in a transaction-driven system, increase the subpool
size until they disappear.

Tuning Comments: Keep these in mind:

- **Use multiple subpools.**
  Multiple subpools minimize contention for the OSAM subpool latch. For very
  busy subpools, this reduction in contention can be measurable.

- **Check whether altered buffer steals occurred in a transaction-driven
  environment.**
  This is a clear indication that a subpool is too small. The solution is to
  arbitrarily increase the size of the subpool until the altered buffer steals
  disappear.

- **Check whether altered buffer steals occurred in IMS batch or a
  non-message-driven (NMD) BMP. If so,**
  - Ensure that the subpool is large enough to eliminate log write-ahead
    (reported as NOT-IWAIT time by IMS monitor).
  - Replace all forced writes with OSAM chained writes.
  - Ensure that sync point or checkpoint frequency is adequate.

When a block is altered, several things happen, especially the following:

- Changes to the block are placed in a log buffer, the block is marked
  altered, and it becomes the most recently used block on the use chain.

- As time goes by, the altered block works its way down the use chain
  toward the least recently used end of the chain as long as it is not
  referenced again. During this period, the log buffers may fill up and be
  written to the log data set or some other activity (particularly with an online
  system) may cause the log buffer in question to be physically written to the
  log device.

- If the altered block remains unreferenced, buffer steal eventually selects the
  buffer to be stolen. When that happens, a log write ahead is requested
  (and steal must wait) unless the data base change log records have
  already been physically written to the log device.

A large enough buffer pool prevents LWAs.

OSAM altered buffer steals of subpools can be eliminated by judiciously
balancing the sizes of the various subpools with timely sync points by
application programs.

### 11.5.3.7 VSAM Buffer Stealing
If no buffers are marked as empty, IMS needs to steal a buffer in order for a CI to
be read into a subpool. The buffer steal algorithm for VSAM is simpler than for
OSAM; a given subpool is searched in two passes rather than three.
First Pass: The first increment to be searched is an arbitrary 40% of the LRU buffers in the subpool. The search criteria are:

- Not owned
- Not altered
- Not busy reading or writing
- Not locked

The first buffer found that meets these criteria is stolen and the search ends.

Second Pass: The entire buffer pool is searched, and the first buffer encountered that is not owned or locked is selected for stealing. With a sensibly sized subpool, this pass is almost certain to find a usable buffer.

Tuning Comments: For transaction-driven systems, make VSAM subpools large enough to avoid:

- Altered buffer steals (forced or space writes). It is best to write all buffers altered by transaction processing to the databases only at sync point time
- Waits for log write-ahead. LWA may occur before an altered buffer steal. If steals are eliminated, then LWA cannot occur.

For IMS batch or non-message-driven BMP activity, the previous comments for avoiding altered buffer steals and LWA are still appropriate. Keep in mind that sync point in the batch and NMD BMP world is caused by application program CHKP and SYNC calls. CHKP and SYNC calls should be frequent enough in conjunction with the size of a subpool such that altered buffer steals and LWA are avoided.

Avoiding buffer rewrites is more applicable to batch rather than transaction processing. Once a transaction takes a sync point, the write of an altered buffer cannot be avoided, as sync point writes out all altered buffers.

In the batch world, the application program has some flexibility as to when altered buffers are written, because it can decide when to issue a CHKP or SYNC call. If it is known that the same database record or segment is changed multiple times, then issuing a CHKP or SYNC call after multiple updates have occurred reduces the number of writes required for a given CI.

For batch and possibly BMP, VSAM's background write function is actually the best way to write out altered VSAM CIs (as opposed to sync point). Background write tends to overlap writes with other processing, whereas sync point does not.
11.5.3.8 Locate Block/CI Within Range

When a segment is to be inserted into an HDAM or HIDAM database, IMS tries to put it in the most desirable block (MDB). However, if insufficient space is available in the MDB, IMS employs the space search algorithm, which looks for a block/CI with sufficient free space. The algorithm executes in the following sequence:

1. Most desirable block.
2. Second most desirable block (using DBD-defined free space).
3. Block/CI in the buffer pool on same cylinder.
4. Block/CI on the same track where bit map indicates space is available.
5. Block/CI on same cylinder where bit map indicates space is available.
6. Block/CI in buffer pool within SCAN= cylinders.
7. Block/CI within SCAN= where bit map indicates space is available.
8. Any block/CI in buffer pool at the end of a data set.
9. Any block/CI at the end of a data set where bit map indicates space is available.
10. Any block/CI in a data set where bit map indicates space is available.

Steps 3, 6, and 8 can involve one or more scans of the buffers in a subpool. IMS uses “Locate Block/CI within Range” for this purpose. If a subpool is large, the buffer handler requests consume a considerable amount of CPU resource.

Space search requests are ideally satisfied from Step 1 (preferred) or Step 2 of the algorithm. Note that Step 2 is invoked only if a free block frequency factor (FBFF) and SEARCHA=0 or 2 are defined in the DBD of the database in question.

The scan processing is invoked for ISRT calls and REPL calls for variable length segments. The scan path length associated with the HD space-search algorithm is directly proportional to the size of the pool and is reported as NOT-IWAIT time by the IMS monitor. The number of VSAM SCHBFRs issued is reported on the IMS monitor VSAM subpool statistics. If the most or second most desirable blocks (Steps 1 and 2 of the algorithm) cannot satisfy a space search request, then pool searching takes place.

Tuning Comments: These may be of use:

- For HIDAM only (not HDAM), allow the use of the second most desirable block. Using the second most desirable block requires the specification of an FBFF and SEARCHA=0 or 2 in the DBD. Both the most and second most desirable blocks are found in a subpool through an efficient hashing technique based on the block/CI RBA.
- Only virtual storage buffers are searched when scanning a subpool in an attempt to satisfy a request for space. For VSAM users, Hiperspace can be used to limit the number of buffers that are searched. For OSAM and VSAM users, implementing multiple smaller subpools also limits the number of buffers searched.
- The best solution is to eliminate or minimize the need for scanning subpools. This can be accomplished by regularly monitoring the databases for information on free space availability and hot spots and by performing timely reorganizations.
### 11.6 IMS Reports and Statistics

At system checkpoint time, IMS logs the buffer pool statistics for OSAM and VSAM. They are also written to the IMS monitor. Therefore, both the IMS monitor print utility and IMSPARS can produce formatted reports for all the database subpools. This section covers the IMS monitor reports; however, the same information is available from the corresponding IMSPARS reports.

#### 11.6.1 IMS Monitor VSAM Buffer Pool Report

Figure 86 shows sample of the IMS monitor VSAM buffer pool report

<table>
<thead>
<tr>
<th>VSAM BUFFER POOL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FIX INDEX/BLOCK/DATA</strong></td>
</tr>
<tr>
<td><strong>SHARED RESOURCE POOL ID</strong></td>
</tr>
<tr>
<td><strong>SHARED RESOURCE POOL TYPE</strong></td>
</tr>
<tr>
<td><strong>SUBPOOL BUFFER SIZE</strong></td>
</tr>
<tr>
<td><strong>TOTAL BUFFERS IN SUBPOOL:</strong></td>
</tr>
<tr>
<td><strong>TOTAL HIPERSPACE BUFFERS IN SUBPOOL</strong></td>
</tr>
</tbody>
</table>

1 RETRIEVE BY RBA CALLS 9,751
2 RETRIEVE BY KEY CALLS 725,131
3 LOGICAL RECORDS INSERTED INTO ESDS 0 → DATA SET EXTENSION
4 LOGICAL RECORDS INSERTED INTO KSDS 0
5 LOGICAL RECORDS ALTERED IN THIS SUBPOOL 9,747
6 TIMES BACKGROUND WRITE FUNCTION INVOKED 8,921 → COUNT MEANS IT WAS ACTIVE
7 SYNCHRONIZATION CALLS RECEIVED 24 → CHKP/SYNC CALLS
8 PERM WRITE ERROR BUFFS NOW IN THE SUBPOOL 0
9 LARGEST NBR OF PERM ERR BUFFS EVER IN THE SUBPOOL 0
10 VSAM GET CALLS ISSUED 853,546
11 VSAM SCHBFR CALLS ISSUED 0 → LOOKING FOR SPACE
12 CONTROL INTERVAL REQUESTED ALREADY IN POOL 712,775 → USED TO CALCULATE HIT
13 CONTROL INTERVAL READ FROM EXTERNAL STORAGE 139,964 → RATIO
14 VSAM WRITES INITIATED BY IMS/ESA 28,630 → BCKGRND WRITE AND CHKP
15 VSAM WRITES TO MAKE SPACE IN THE POOL 8,920 → ALTERED BUFFER STEALS
16 VSAM READS FROM HIPERSPACE BUFFERS 0 → HIPERSPACE
17 VSAM WRITES TO HIPERSPACE BUFFERS 0 → EFFECTIVENESS
18 FAILED VSAM READS FROM HIPERSPACE BUFFERS 0 → EXPANDED STORAGE
19 FAILED VSAM WRITES TO HIPERSPACE BUFFERS 0 → OVERCOMMITTED

**TOTAL I/O OPERATIONS** 177,514

*Figure 86. VSAM Buffer Pool Example*
11.6.1.1 Report Analysis

Figure 86 on page 265 is a VSAM Buffer Pool report for a subpool with 20 buffers, each 1 KB in size. Comments on the right side of the figure are added for your reference and use. Also, each line item in the report is numbered for easy reference.

- **Line 3, Logical records inserted into ESDS**
  Counts in this field indicate the IMS-maintained logical EOF marker has been moved to extend one or more ESDSs. In VSAM terms, the EOF is moved one CA at a time. These counts indicate that one or more data set are extended to satisfy requests for space as a result of inserts or variable length replaces where the replaced segment had to be split. One or more databases are likely to require reorganization to regain free space. Database DBDs may need to be changed to increase free space specifications.

- **Line 6, Times background write function invoked**
  Counts in this field indicate that background write is active. If the count field is zero, background write may or may not be turned off. If the count field is zero and altered buffer steals are occurring (see line 15 of Figure 86 on page 265), it is likely that background write is turned off. Background write is activated by control card input (//DFSVSAMP DD statement in batch or DFSVSMxx member for online systems). By default, background write is active. If turned off, we recommend you turn it on (see 11.3.3, “VSAM Background Write” on page 243).

- **Line 7, Synchronization calls received**
  This count is incremented for every transaction sync point and every BMP CHKP and SYNC call. Only when every transaction updates buffers in a subpool does the count match the number of buffer purge calls.

- **Line 11, VSAM SCHBFR calls received**
  SCHBFR is an IMS macro that may be translated as search buffer. A large number indicates the subpool is frequently being scanned in a search for space. If this is a batch monitor, the VSAM statistics report will identify which database data sets are incurring the SCHBFR calls and, thus, are good candidates for reorganization or respecification of free space.

  **Note**
  
  An ISRT call typically causes multiple SCHBFR requests to be issued. Even when SCHBFR finds a buffer in the subpool, IMS examines it to see if there is sufficient room for the segment. If not, another SCHBFR is issued to continue the scan of the subpool.

- **Line 12, Control interval requested already in pool**
  This count indicates the number of times a request for a CI was satisfied from the subpool (that is, a read operation was not required). It is used to calculate a hit ratio for the subpool (discussed under “Statistics” on 268 at the end of this list).

- **Line 13, Control interval read from external storage**
  The count reports the number of reads from external DASD required to satisfy requests made of the buffer handler. In other words, these requests could not be satisfied from the contents of the subpool and had to be read in from a
database data set on the DASD. Use tuning tactics to attempt to eliminate reads or, if they cannot be avoided, to perform the reads in as efficient a manner as possible. This line item is used to calculate a hit ratio for the subpool (discussed in 11.6.1.2, “Statistics” on page 268).

- **Line 14, *VSAM writes initiated by IMS/ESA***
  The count reports the number of CIs written to database data sets as a result of application program sync points (including CHKP and SYNC calls) and background write.

- **Line 15, *VSAM writes to make space in the pool***
  The count reflects the number of CIs written to database data sets as a result of altered buffer steals. You can eliminate altered buffer steals entirely by judiciously balancing the number of buffers of the pool with timely application program sync points and background write activity.

- **Lines 16 and 17, *VSAM reads from Hiperspace buffers* and *VSAM writes to Hiperspace buffers***
  The reads (Line 16) reflect a request of the buffer handler that was satisfied from Hiperspace. The writes (Line 17) reflect the movement of CIs from the virtual storage subpool buffers to Hiperspace. Every read from DASD or Hiperspace into the virtual storage subpool results in a write to Hiperspace.

  The ratio of writes and reads to and from Hiperspace is an indication of its effective use. Hiperspace access (read or write) consumes processor cycles. A write:read ratio of around 4:1 or 5:1 is equivalent to the path length required for an actual read from DASD. A high write to read ratio, 20:1 for example, avoids a reread 5% of the time and consumes four or five times the amount of processor required if each of the reads is from DASD.

  Reasons for using Hiperspace rather than virtual storage buffers include:
  
  - A VSAM subpool in virtual storage is limited to 32 KB buffers, But there is no practical limit to the number of Hiperspace buffers that can be used. Therefore, if buffers larger than 32 KB are required, then Hiperspace is the only way to obtain them.
  
  - From a buffer handler point of view, SCHBFR (search buffer operations, Line 11) requests consume CPU resource; the more buffers in a subpool, the longer the path length. SCHBFR requests do not search Hiperspace buffers. Therefore, if SCHBFR requests are perceived to be a problem, placing most of the buffers in Hiperspace will reduce the path length required to satisfy the requests. For HIDAM, a better solution is to provide enough free space so that the most desirable and second-most desirable CIs have enough space to satisfy requests and then to schedule timely database reorganizations to buy back that free space when required.

  VSAM reads from Hiperspace buffers is used to calculate a hit ratio for the subpool (discussed in 11.6.1.2, “Statistics” on page 268).

- **Lines 18 and 19, *Failed VSAM reads from Hiperspace buffers* and *Failed VSAM writes to Hiperspace buffers***
  Counts in either of these two fields indicate that expanded storage is overextended. Hiperspace, as implemented by VSAM, does not guarantee that Hiperspace buffers are not lost. Expanded storage is managed by the MVS storage manager. If the MVS storage manager concludes that expanded storage is overcommitted it can, and will, steal Hiperspace buffers for other
uses. Counts in these two fields reflect Hiperspace buffer steals by the MVS storage manager.

### 11.6.1.2 Statistics

Several statistics are of use when analyzing a VSAM database subpool. The following are some of the more common statistics used:

- **Hit Ratio**

  A subpool hit ratio is calculated as follows:

  \[
  \text{Hit Ratio} = \frac{\text{Line 12} + \text{Line 16}}{\text{Line 12} + \text{Line 13} + \text{Line 16}} \times 100 = \text{xx.x}\%
  \]

  For this example, the hit ratio calculation is as follows:

  \[
  \text{Hit Ratio} = \frac{712,775 + 0}{712,775 + 139,964 + 0} \times 100 = 83.6\%
  \]

  By itself, a hit ratio means nothing. Hit ratios are generally useful when making run-to-run comparisons. If a change is made to a particular subpool and the hit ratio increases, it is very likely that the increased hit ratio represents an improvement.

- **Reads per Second** and **Writes per Second**

  Reads per Second are simply “Control interval read from external storage” (line 13) divided by the monitor interval in seconds. The tuning goal is to reduce this number.

  Writes per Second are “VSAM writes initiated by IMS/ESA” (Line 14) plus “VSAM writes to make space in the pool” (line 15), all divided by the monitor interval in seconds. The tuning goal is to reduce this number.

- **Buffer Life**

  This statistic is useful for online systems. It is calculated as follows:

  \[
  \text{Buffer Life} = \frac{\text{No. of buffers in subpool} \times \text{Monitor Interval}}{\text{CONTROL INTERVAL READ FROM EXTERNAL STORAGE (Line 13)}}
  \]

  Buffer life is the average life in seconds of the contents of a buffer in a subpool. When the average buffer life is in tens of seconds, then the subpool is holding buffer contents across a terminal user's think time. When buffer life is low, such as less than 1 buffer-second per read, it is generally beneficial to increase the number of buffers in a subpool dramatically.

  Use buffer life in conjunction with reads per second. For example, if the number of buffers in a subpool is increased, the goal is for buffer life to increase and reads per second to decrease. If reads per second do not decrease, then the increase in the number of buffers is not beneficial and not worth the extra demand placed on real storage by the additional buffers.
11.6.2 IMS Monitor OSAM Buffer Pool Report

Figure 87 on page 270 shows the IMS monitor OSAM buffer pool report.

11.6.2.1 Report Analysis

Figure 87 on page 270 is an OSAM buffer pool report for a subpool with 12 buffers, each 12 KB in size.

Comments are provided on the right side of the figure for your reference and use. Also, each line item in the report is numbered for easy reference.

- **Line 2, Requests to create new blocks**
  Each count represents a format logical cylinder operation that extended a database data set by one physical DASD cylinder. When present, these counts indicate that one or more data sets are being extended to satisfy requests for space as a result of inserts or variable-length replaces where the replaced segment had to be split. One or more databases are likely to require reorganization to regain free space. Database DBDs may need to be changed to increase free space.

- **Line 4, Purge calls**
  Each count means the database subpools were purged of altered buffers for a dependent region or batch job step. The count is typically incremented for transaction sync points, CHKP, and SYNC calls.

- **Line 5, Locate-type calls, data already in subpool**
  Each count reflects the number of times a request for a block was satisfied from the subpool; that is, a read operation was not required.

  This line item is used to calculate a hit ratio for the subpool (discussed in 11.6.2.2, “Statistics” on page 271).

- **Line 7, *Read I/O requests**
  The count reported is the number of reads from external DASD required to satisfy requests made of the buffer handler. In other words, requests that could not be satisfied from the contents of the subpool required that a block be read from a database data set on DASD. Use tuning tactics to attempt to eliminate reads or, if they cannot be avoided, to perform them as efficiently as possible.

  This line item is used to calculate a hit ratio for the subpool (discussed in 11.6.2.2, “Statistics” on page 271).

- **Line 8, *Single block writes by buffer steal rtn**
  The count reflects the number of blocks written to database data sets as a result of altered buffer steals. You can eliminate altered buffer steals entirely by judiciously balancing the size (number of buffers) of the pool with timely application program sync points.

  Writes to OSAM database data sets are accomplished in one of two ways:
  - Writes of single blocks at a time by altered buffer steal
  - Writes of multiple blocks at a time by purge.
*** OSAM DATA BASE BUFFER POOL STATISTICS ***

<table>
<thead>
<tr>
<th>FIX BLOCK/DATA</th>
<th>N/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBPOOL ID</td>
<td></td>
</tr>
<tr>
<td>SUBPOOL BUFFER SIZE</td>
<td>12,288</td>
</tr>
<tr>
<td>TOTAL BUFFERS IN SUBPOOL</td>
<td>12</td>
</tr>
</tbody>
</table>

1 LOCATE TYPE CALLS 8,324,866
2 REQUESTS TO CREATE NEW BLOCKS 1 ➔ FORMAT LOGICAL CYLINDER
3 BUFFER ALTER CALLS 2,531,675
4 PURGE CALLS 24 ➔ SYNC OR CHKP ACTIVITY
5 LOCATE-TYPE CALLS, DATA ALREADY IN SUBPOOL 8,097,059 ➔ USED TO CALCULATE HIT
6 BUFFERS SEARCHED BY ALL LOCATE-TYPE CALLS 8,690,009 ➔ RATIO
7 *READ I/O REQUESTS 116,216
8 *SINGLE BLOCK WRITES BY BUFFER STEAL RTN 115,674 ➔ ALTERED BUFFER STEALS
9 *BLOCKS WRITTEN BY PURGE 185 ➔ BLOCKS WRITTEN BY CHKP
10 TOTAL NBR OF I/O ERRORS FOR THIS SUBPOOL 0
11 BUFFERS LOCKED DUE TO WRITE ERRORS 0
12 LOCATE CALLS WAITED DUE TO BUSY ID 0 ➔ BH CONTENTION
13 LOCATE CALLS WAITED DUE TO BUFR BUSY WRITE 0 ➔ BUFFER POOL TOO SMALL
14 LOCATE CALLS WAITED DUE TO BUFR BUSY READ 0 ➔ SHOULD NOT HAPPEN. WAIT FOR LEVEL 9 OR 10.
15 BUFR STEAL/PURGE WAITED FOR OWNERSHIP RLSE
16 BUFFER STEAL REQUEST WAITED FOR BUFFERS

TOTAL I/O OPERATIONS 232,075

Figure 87. OSAM Buffer Pool Statistics

Purge is much more efficient than altered buffer steal; very measurable performance improvements can be obtained by eliminating all altered buffer steals with OSAM.

- **Line 9, *Blocks written by purge**

  Because OSAM purge is the most efficient way to write altered OSAM buffers, the tuning tactic is to write all altered buffers with OSAM purge and eliminate all OSAM altered buffer steals.

- **Buffer handler contention** is represented by Lines 12 through 16. Contention means a request of the buffer handler waits until the condition indicated is reset. These waits, from an IMS Monitor perspective, are reported as NOT-IWAIT time.
  - **Line 12, Locate calls waited due to busy ID**

    Each OSAM buffer has a buffer prefix. Whenever a prefix must be changed (to indicate a changed attribute, for example), it is given an attribute of BUSY ID under the subpool latch. Once BUSY ID is set, no other task can access the prefix or its associated buffer contents. Any requests for the buffer prefix while it is busy must wait.

  - **Line 13, Locate calls waited due to BUFR busy write**

    A request is made to access the contents of a buffer, but the contents of the buffer are in the process of being written to the proper database data...
set. The write must complete before any requestors can have access to the contents of the buffer.

- Line 14, **Locate calls waited due to BUFR busy read**

The explanation is the same as for Line 13, except the buffer is busy because a read is in progress (a previous requestor required the same OSAM block). BUFR BUSY READ may be thought of as a good form of contention, in that a single read can satisfy multiple requests of the buffer handler.

- Line 15, **BUFR steal/purge waited for ownership rlse**

The OSAM steal or purge routine cannot steal or purge a buffer that is owned by another requestor. Steal or purge must wait until ownership is released.

- Line 16, **Buffer steal request waited for buffers**

The buffer steal routine is required to steal a buffer to satisfy a read request. Counts indicate that the steal routine went through its complete algorithm and could not find a suitable buffer to steal. The only types of buffers that are not suitable for stealing are those with a use level of 9 (buffer has a new ID pending situation) or 10 (buffer is locked because of a write error).

The overall probability of not finding a buffer suitable for stealing is very remote given an adequate number of buffers in a subpool. Therefore, a value of other than zero for “Buffer steal request waited for buffers” is highly unlikely.

### 11.6.2.2 Statistics

Several statistics are of use when analyzing an OSAM database subpool. The following are some of the more common statistics used:

- **Hit Ratio**

  A subpool hit ratio is calculated as follows:

  \[
  \text{Hit Ratio} = \frac{\text{Line 5}}{\text{Line 5} + \text{Line 7}} \times 100 = \text{xx.x}\%
  \]

  For the example in Figure 87 on page 270, the hit ratio calculation is as follows:

  \[
  \text{Hit Ratio} = \frac{8,097,059}{8,097,059 + 116,216} \times 100 = 98.6\%
  \]

  By itself, a hit ratio means nothing. Hit ratios are generally useful when making run-to-run comparisons. If a change is made to a particular subpool and the hit ratio increases, it is very likely that the increased hit ratio represents an improvement.

- **Reads per Second** and **Writes per Second**

  Reads per second are simply “Read I/O requests” (Line 7) divided by the monitor interval in seconds. The tuning goal is to reduce this number.
Writes per second are “Single block writes by buffer steal rtn” (Line 8) plus “Blocks written by purge” (Line 9), all divided by the monitor interval in seconds. The tuning goal is to reduce this number.

- **Buffer Life**

  This statistic is useful for online systems. It is calculated as follows:

  \[
  \text{Buffer Life} = \frac{\text{No. of buffers in subpool} \times \text{Monitor Interval}}{\text{READ I/O REQUESTS (Line 7)}}
  \]

  Buffer life is the average life in seconds of the contents of a buffer in a subpool. When the average buffer life is in tens of seconds, then the subpool is holding buffer contents across a terminal user’s think time. When buffer life is low, such as less than 1 buffer-second per read, it is generally beneficial to increase the number of buffers in a subpool dramatically.

  Use buffer life in conjunction with reads per second. For example, if the number of buffers in a subpool is increased, the goal is for buffer life to increase and reads per second to decrease. If reads per second do not decrease, then the increase in the number of buffers is not beneficial and not worth the extra demand placed on real storage by the additional buffers.

### 11.6.3 DB Monitor Program I/O Report (for Batch DL/I)

This report (Figure 88 on page 273) shows us which database data sets are important to tune by DDNAME within DBD. The database subpools to tune are those in which these DDNAMEs reside.

On the left side of the figure are the PCBNAMEs associated with the PSB that this batch job step was executing against. If background write was active, one of these PCBNAMEs is named and reported as BKGWRPCB. Incidentally, background write activity is reported only for stand-alone batch; it is not reported by an online monitor.

The presence of BKGWRPCB means that background write is active and was not turned off. This fact should not be ignored. Ignore, however, the statistics associated with the background write PCB. In terms of job step elapsed time, BKGWRPCB does not contribute to the elapsed time of the job step. On the lower right side of the 88, this adjustment has been made.

So what should be tuned? The database DDNAMEs that contribute most to the elapsed time of the job step are the ones to be analyzed and tuned. These usually can be easily spotted by scanning the IWAIT TIME, TOTAL column. In Figure 88 on page 273, DISTR. NOS. 91, 94, 92, and 93 (the top four in terms of elapsed time) account for 9,371 seconds out of a total job step IWAIT time of 10,758 seconds, or 87% of the total. From the perspective of a database buffer handler, these four DDNAMEs represent 526,735 IWAITs, or 85% of all IWAITs for the job step. Give these four DDNAMES high priority for analysis and tuning actions related to the I/O subsystem and identify the specific data base subpools for analysis and tuning actions.

Note that the DDNAMEs are selected for analysis and tuning without any regard for average IWAIT time. Only after the DDNAMEs are selected do you look at mean IWAIT time as part of the analysis phase of the analysis and tuning effort.
### Program I/O Report Example

**Table 88**

<table>
<thead>
<tr>
<th>PCBNAME</th>
<th>IWaits</th>
<th>Total</th>
<th>Mean</th>
<th>Maximum</th>
<th>DDNAME</th>
<th>Module</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD09DP</td>
<td>133882</td>
<td>213315510</td>
<td>15934</td>
<td>1169849</td>
<td>DBD09DP</td>
<td>DBH</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>82500</td>
<td>1537020964</td>
<td>18630</td>
<td>901899</td>
<td>DBD09DI</td>
<td>VBH</td>
<td>94</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td>216382</td>
<td>3670336474</td>
<td>16962</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBDO0DP</td>
<td>239432</td>
<td>4733542430</td>
<td>19769</td>
<td>993914</td>
<td>DBDO0DP</td>
<td>DBH</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>13122</td>
<td>212103304</td>
<td>16163</td>
<td>598764</td>
<td>DBDO054</td>
<td>VBH</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>32504</td>
<td>456751087</td>
<td>14052</td>
<td>1210959</td>
<td>DBDO052</td>
<td>VBH</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>7130</td>
<td>106784880</td>
<td>14976</td>
<td>565013</td>
<td>DBDO053</td>
<td>VBH</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>28955</td>
<td>394279638</td>
<td>13616</td>
<td>804640</td>
<td>DBDO051</td>
<td>VBH</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>4909</td>
<td>90898710</td>
<td>18516</td>
<td>416125</td>
<td>DBDO050</td>
<td>VBH</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>27011</td>
<td>13505</td>
<td>16033</td>
<td>DBDO056</td>
<td>VBH</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>46439</td>
<td>9327</td>
<td>13258</td>
<td>DBDO055</td>
<td>VBH</td>
<td>109</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td>326059</td>
<td>5994433699</td>
<td>18384</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD07DP</td>
<td>79021</td>
<td>967217484</td>
<td>13637</td>
<td>617579</td>
<td>DBD07DP</td>
<td>DBH</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>6337</td>
<td>117046057</td>
<td>18470</td>
<td>110672</td>
<td>DBD07DI</td>
<td>VBH</td>
<td>96</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td>77258</td>
<td>1084263541</td>
<td>14034</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BKGWRPCB</td>
<td>16640</td>
<td>623198401</td>
<td>37451</td>
<td>964698</td>
<td>DBD09DI</td>
<td>VBH</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>3666</td>
<td>118906069</td>
<td>32433</td>
<td>1796075</td>
<td>DBD0051</td>
<td>VBH</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>1077</td>
<td>41807551</td>
<td>38818</td>
<td>1675754</td>
<td>DBD0701</td>
<td>VBH</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>1308</td>
<td>48676758</td>
<td>37215</td>
<td>404777</td>
<td>DBD0054</td>
<td>VBH</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>3781</td>
<td>154384410</td>
<td>40831</td>
<td>1638536</td>
<td>DBD0052</td>
<td>VBH</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>770</td>
<td>29781128</td>
<td>38676</td>
<td>989841</td>
<td>DBD0001</td>
<td>VBH</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>970</td>
<td>30801392</td>
<td>31754</td>
<td>679660</td>
<td>DBD0053</td>
<td>VBH</td>
<td>107</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td>28212</td>
<td>1047551149</td>
<td>37131</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 88. Program I/O Report Example**

### 11.6.4 IMS Monitor DL/I Call Summary Report

This report (Figure 89 on page 274) can be used to find out where the job step is spending most of its time in general (IWAIT time versus NOT-IWAIT time) and where the job step is spending most of its time in terms of particular DL/I calls.

**IWAIT or NOT IWAIT:** The batch total line reveals in general where the job step is spending its time. The average elapsed time for all DL/I calls is 4.73 ms and, of this, 1.36 ms per call is spent on NOT-IWAIT time. The difference between the two, 3.37 ms (4.73 ms minus 1.36 ms), represents the IWAIT portion of the average DL/I call for this job step. This 3.37 ms of IWAIT time per call is 71% of the overall average DL/I call time. Clearly, the tuning of this job step requires that priority be given to tuning IWAIT time versus NOT-IWAIT time.

This sort of general analysis can also help to set tuning expectations. That is, if only IWAIT time is addressed, DL/I call elapsed time cannot be cut by more than 71%.
### DL/I Calls to Analyze:
From a buffer handler perspective (database reads and writes), one algorithm for selection is to select those DL/I calls that incurred the greatest number of IWAITS. From the example, the calls on lines 5, 6, and 17 are appropriate for selection.

Given these three calls, one approach is to lay out the physical DBD structure and note the database data set DDNAMEs that must be accessed to satisfy each call. It is necessary to understand the application program logic that issues these calls. Creativity and knowledge are needed to analyze and take tuning actions related to these three calls. The key resources for these calls are likely to have been identified from studying the Program I/O report. The DDNAMEs and subpools of interest for analysis and tuning actions come from this report.

### NOT-IWAIT Time:
NOT-IWAIT time accounts for 29% of the average elapsed time for a call. The buffer handlers contribute to NOT-IWAIT time in terms of path length and buffer handler contention (OSAM buffer handler). Listed below are some factors to consider when performing analysis and tuning:

- Path length
  - Look at the subpool reports associated with the four selected DDNAMEs:
- Can reads and writes be eliminated? Path length and elapsed time are saved for every read or write eliminated.

- Are database data sets being extended? If so, this indicates that the most desirable and second most desirable blocks or CIs do not have enough free space. Lack of free space results in extra buffer handler (and action module) path length to search for space using the HD space search algorithm.

- Is Hiperspace in use? What is the write-to-read ratio? If it is too high, path length may be wasted.

- Are there a large number of altered buffers? The stealing of altered buffers requires path length. Altered buffer steals can be eliminated.

  - Buffer handler contention (OSAM). Do the OSAM subpool reports show contention? Separate subpools for busy OSAM database data sets can reduce this contention.

  - Buffer handler resources

    - What are the statistics for the subpools associated with the four selected DDNAMEs? Arbitrarily increasing the number of buffers can eliminate rereads and perhaps reduce the number of writes.

    - Is the block or CI size large enough? An increased block or CI size always reduces the read and write activity of sequential readers and updaters.

    - Can sequential readers and writers be split into separate subpools? Sequential readers and updaters usually do not require large subpools. Nevertheless, they can dominate a subpool to the detriment of other DDNAME read and write activity. Splitting sequential readers and writers into separate subpools not only saves buffer handler pathlength, but it also reduces read and write activity.

### 11.6.5 Database Monitor VSAM Statistics

Figure 90 on page 276 shows the statistics generated for stand-alone batch IMS Monitor reports.
The statistics shown in Figure 90 are generated for stand-alone batch IMS monitor reports only. The VSAM statistics report complements the VSAM buffer pool reports. The column headings in the VSAM statistics report are the line items in a VSAM buffer pool report.

The column labeled SCHBFR can lead to the DDNAMEs that are causing the buffer handler to search for space beyond the most and second-most desirable CIs. In other words, this report can show exactly which databases are candidates for reorganization and which DBDs to analyze for free space specification and reorganization frequency.

The leftmost three columns of the report show generic DL/I call functions within DDNAME within DB PCBNAME. This is frequently useful for analysis of the DL/I calls selected from the call summary report. As can be seen, for any particular type of call, the IWAITs by DDNAME for that type of call are listed.

### 11.6.6 IMS /Display Pool Command

IMS monitor reports are generated for the time period that the monitor was run and must be printed offline using the monitor print program (DFSUTR20). It is possible to view the statistics for database buffer pools as they stand at a point in time, using the /DISPLAY POOL command. This command is appropriate for both full-function and Fast Path buffer pools.
11.6.6.1 /DISPLAY POOL DBAS

The full function database buffer pools are displayed using the /DISPLAY POOL DBAS command. The output of this command is shown in Figure 91 on page 278.

The output of the /DIS POOL DBAS command shows statistics on both the OSAM and VSAM buffer pools.

**OSAM Buffer Pools:** The first section of the display reports on the status of the OSAM pools, first indicating whether sequential buffering is being used, then giving a report on each OSAM subpool before reporting on the OSAM pool as a whole. The size of each subpool can be obtained by multiplying the block size (BSIZE) by the number of buffers (NBUF). Each of the individual fields in the report of each subpool is the same as the lines in the report of the IMS Monitor. For example, the abbreviation LCTREQ corresponds to the number of locate type calls found in the IMS monitor OSAM buffer pools report.

**VSAM Buffer Pools:** Each VSAM subpool is reported on, in order of CI size. This is followed by a report on the total VSAM buffer pool, summarizing all the subpool reports. The fields in the report match the report of the IMS monitor. The last two lines on each of the subpool and total buffer reports indicate the use of the VSAM Hiperspace facility. The lines show the number of reads (HSR-S), the number of writes (HSW-S), the total number of Hiperspace buffers (HS NBUFS), and the number of read/write failures and errors on the Hiperspace buffers.

The /DISPLAY POOL DBAS command offers a quick and easy way of observing the activity of the various OSAM and VSAM subpools.

11.6.6.2 /Display Pool FPDB

Fast Path buffer pool information is displayed using the /DISPLAY POOL FPDB command. The output of this command is shown in Figure 92.

<table>
<thead>
<tr>
<th>FPDB BUFFER POOL</th>
<th>AVAIL = 60</th>
<th>WRITING = 0</th>
<th>PGMUSE = 0</th>
<th>UNFIXED = 90</th>
</tr>
</thead>
</table>

*Figure 92. IMS Command /DIS POOL FPDB*

The four values reported are:

**AVAIL:** Shows the number of available page-fixed buffers.

**WRITING:** Indicates the number of buffers waiting to be written to disk or being written.

**PGMUSE:** Shows the number of buffers that are page fixed because of dependent region NBAs, and the largest OBA.

**UNFIXED:** The number of free buffers still available for additional regions to acquire (NBA), or for areas with SDEPs to be started.
### Sequential Buffering

<table>
<thead>
<tr>
<th>Buffer Pool</th>
<th>ID</th>
<th>Size</th>
<th>Size</th>
<th>NBUF</th>
<th>FX</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSAM DB</td>
<td>D</td>
<td>4K</td>
<td>1</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM DB</td>
<td>VLP1</td>
<td>2K</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM DB</td>
<td>VLP1</td>
<td>4K</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM DB</td>
<td>VLP1</td>
<td>8K</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM DB</td>
<td>BSIZE</td>
<td>ALL</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 91

IMS Command /DIS POOL DBAS
Chapter 12. IMS Application and Database Design for Performance

This chapter focuses on database and application design factors that might contribute to IMS performance problems. Consideration of these factors in the design stages results in a more efficient IMS system.

12.1 The Importance of Design Issues

IMS application design is responsible for system performance in that it is the application program that controls the number of DL/I calls and, for the most part, the number of database I/Os per call. Application programs are, more often than not, the largest contributor to performance problems. It is also true that application programs are the most difficult area of an IMS system to tune. An application change may not be simple to accomplish, and the impact of the change can be far reaching. The ideal time to remove performance bottlenecks from databases and application programs is in the design stage, before any application coding has occurred.

The IMS system is designed to allow multiple applications to share the same resources. One poorly designed application can impact the throughput and response time of the entire IMS. The objective of good IMS application design is to have a transaction complete its unit of work in the minimum number of DL/I calls and I/Os and to allow the resources that it holds to be freed as quickly as possible, making them available to the next transaction.

It may be necessary to have a transaction that issues hundreds of DL/I calls for a unit of work. This naturally results in a longer response time for the transaction, but it may also cause longer response times for other transactions that are waiting for the resources this transaction has tied up.

Similarly, a poorly designed database can have a major impact on many transactions and hence on the whole system. Tuning a database to improve overall performance can sometimes be achieved without any application changes (this is the beauty of data independence). An example might be where the hierarchic access method is changed from HISAM to HDAM, or where a physical-child-last pointer is added to a segment.

Some changes, however, can be made only in conjunction with application program changes. These are most undesirable, so invest as much effort as needed into getting the database design right the first time.

12.2 Database Design

From a performance perspective, database design may be thought of as a trade-off between reducing the number of segment types in a database and gaining the benefits of data independence. Database design inadequacies can result in additional database I/Os per call and contention for database records between application programs.
The reduction in the number of database segments allows for better transaction performance by reducing the number of I/Os required to access data. On the other hand, the separation of the data into additional segments based on characteristics and relationships to other data leads to better granularity and greater data independence. Data independence is the characteristic that allows for cost and time savings in application maintenance, ease of development of future systems, and the integrity of the data that IMS is renowned for.

The greatest performance improvements can be realized by focusing on the databases that account for the majority of I/O activity. The 80/20 rule applies here; it is often found that 80% or more of all database I/O is attributable to 20% or less of all the databases. The 20% or so most active databases need to be identified, and this is where the real benefit will be found. Other databases can be checked as well, but improvements to them will have less impact.

### 12.2.1 Access Methods

In systems where performance is a primary concern, consider DEDB as the first choice of access method, with HDAM second. The reason for this choice is that most data requests to retrieve the root directly can be satisfied with one I/O as opposed to two to three I/Os for the indexed methods. Improving online performance by reducing I/Os is, however, not the only consideration in selecting an access method. The other access methods are not to be discarded, because, in some environments, their advantages may outweigh those of DEDB and HDAM.

#### 12.2.1.1 Data Entry Database

DEDBs are HDAM-like databases with significant performance and availability enhancements. For the most part, DEDBs provide the same functionality as HDAM databases with the exception that secondary indexes and logical relationships are not supported for DEDB. The DEDB is designed for performance. A special segment type, the sequential dependent segment (SDEP), is specifically designed for very efficient data collection, data entry, journaling, and other activities. The SDEP segments are not written to DASD individually, but are accumulated in special buffers that are written out only when completely full. Most DEDB processing is done in parallel to allow multithreading.

DEDBs can be partitioned into what are known as areas. Areas are independent of one another. If for some reason an area is unavailable, processing continues for the others. A transaction accessing an unavailable area is still scheduled, and any calls to records in that area receive an FH status code.

#### 12.2.1.2 Hierarchic Direct Access Method

Undesirably long HDAM synonym chains can occur in databases where the root addressable area is too small or where not enough root anchor points (RAPs) are available per block or CI. Continuous monitoring shows a gradual increase in I/Os per call as the chains grow. It may also show an increase in PI waits, because with HDAM, locks are taken on the RAP rather than on the root. If these chains grow because the root addressable area is too small, reorganize the database, and reload it with a larger root addressable area allocated to the database. HDAM physical sequence sort/reload (PSSR), a part of the DBT program product, can be used to vary certain DBD parameters. The output can then be analyzed to see the effect on the number of roots that have randomized to each RAP and the number of roots that have randomized to each block. PSSR can use the output of the HD unload program or the DBT high speed sequential retrieval (HSSR) as input.
PSSR can provide the following tuning analysis reports:

- DBD parameters and overrides
- Assigned roots per RAP
- Assigned roots per block

The HD pointer checker can be used to analyze the actual number of roots that have randomized to each RAP and the number of roots that have randomized to each block.

Specifying free space for the HDAM primary data set group is self-defeating. It creates a cascading effect on the root addressable area by forcing roots that randomize to a particular block and RAP to be stored in a location other than the most desirable block. Consider creating a separate data set group with a free-space specification different from that of the primary data set group for dependent segments that are frequently inserted or extremely volatile.

Consider the use of a sequential randomizer with HDAM for those databases that require quick random access along with good sequential processing techniques.

In the DBD, the bytes operand of the RMNAME parameter can be used in HDAM to limit the number of dependent segments allowed in the root addressable area at initial load time or during a series of ISRT calls not broken by a call to another database record. This can be extremely efficient when the database has many segment types but most of the processing is against the root segment or segments that are physically located close to the root and reside in the root addressable area.

Load large HDAM databases in physical block sequence. This reduces the arm and head movement on the DASD, thereby reducing the time required to load the database. The randomizer can be invoked as a presort exit so that the load program input file can be sorted on record key within RAP within block, which then corresponds exactly with the HDAM data set order.

This same technique can be used for many HDAM-updating jobs as well.

**HDAM Randomizing Routines:** Make HDAM randomizers reentrant\(^{19}\) so that they can be used by multiple DL/I calls in parallel. This prevents bottlenecks when a heavily used database is accessed by many dependent regions at the same time or when one randomizer is being used for many databases.

---

\(^{19}\) IBM-supplied randomizers (for example DFSHDC40) are all reentrant.
The following are the link-edit attributes with their corresponding impact on randomizers:

- **REENTRANT** Provides full parallel access.

- **REUSABLE** A latch is taken on the database before calling the routine. If the routine is used for multiple databases, then it must be written and compiled as REENTRANT.

- **NONREUS** A latch is taken on the database before calling the routine, and if it is for multiple databases, then each database must have its own copy of the routine.

### 12.2.1.3 Hierarchic Indexed Direct Access Method
HIDAM can be very efficient when processing database records in root key sequence, particularly when complete database records fit in a single block/CI or if the database is root only. For example, HIDAM might be used in lieu of HDAM for a database that is frequently processed in batch in key sequential order.

The use of PTR=TB must be specified for HIDAM roots if root key sequential processing is important. It allows GN processing that is not key qualified to proceed along the physical twin forward chain without reference to the HIDAM index. Note that a GN call for a root with a key qualification that cannot be satisfied at the current position always proceeds through the index.

### 12.2.1.4 Hierarchic Indexed Sequential Access Method
If the database contains nonvolatile segments of approximately the same size and requires sequential processing, consider the use of HISAM. The I/O to randomly retrieve a HISAM root (KSDS access) can be faster than that to retrieve a HIDAM root (KSDS and ESDS access). HISAM database loads and utilities are also faster than the HD equivalents.

Consider simplify-HISAM using native VSAM processing when frequent DL/I batch sequential jobs are required and the database is root only.

### 12.2.2 Database Structure and Segmentation
Keep database structures as simple as possible. Complex structures can have too many segments, resulting in an excessive number of database calls. If a segment occurs a limited number of times under its parent or if the length of a segment prefix exceeds the length of the data in the segment, consider an alternative design. For example, consider allocating sufficient space for the maximum number of occurrences and storing this data in the parent segment itself, combining all dependent segments into the parent. If the smaller segment is a parent segment, consider moving its fields into the lower level segments and expanding the key field, particularly if the data is static or contains key data only. When making these changes, the reduction in database I/O significantly outweighs the disadvantages of additional maintenance to keep the duplicate data in sync and the reduction in data independence.

Keep the most frequently used segments as close to the root as possible. Ideally, the most frequently accessed data is stored in the root. This is often not realistic or practical because of the need to group data into multiple segments. Make every attempt to keep these segments at a high segment level and in the left-most of the hierarchy, minimizing the probability of I/O when accessing the segment through the root.
12.2.3 Long Physical Twin Chains

Long physical twin chains can show up in the IMS monitor DL/I call summary report as many I/Os per call (depending on length of chain) and high DL/I NOT-IWAIT time. Plausible solutions to reduce chain chasing are to restructure the database or access the database through a secondary index. Database restructuring involves breaking up the long twin chain by using dummy parent segments. The key to these dummy parent segments is a partial key to the children who are dependent upon it. If the dummy segments are very short, many of them can fit into one block of data. Therefore, a further reduction in I/Os can realized by placing these dummy segments in their own data set group. If the decision is to create a secondary index, refer to 12.2.8, “Logical Relationships and Secondary Indexing” on page 286 for performance information relating to secondary indexes.

A further alternative to long physical twin chains is the use of subset pointers in a DEDB. Subset pointers provide a means of searching a twin chain, beginning at a planned point in the chain.

12.2.4 Database Reorganization

The indications of when to reorganize databases can be established by various methods. For OSAM data sets, regularly review the extent and the last block pointer information from a VTOC listing. For VSAM ESDSs, review the high-use RBAs from the VSAM LISTCAT utility. When the last block pointer or high-use-RBA begins to increase or when new extents keep being created, you have a good indication that segments are being added to the end of the data set, which in turn can indicate the need for database reorganization. For VSAM KSDSs, the need for reorganization is indicated by an increase in CI or CA splits as indicated in the VSAM catalog.

The need for database reorganization may also be indicated by an increasing number of I/Os per call as shown by the IMS monitor DL/I call summary report (Figure 36 on page 104). This trend shows itself on successive monitor runs. The solution is to reorganize the database. Database reorganization is a slow process, so large databases should be designed to need only infrequent reorganization. The use of free space in data sets where database segments are frequently inserted reduces the need for frequent reorganization. Data set groups can be used to isolate volatile segments—or segments with widely varying lengths—from the remaining database segments. The advantage of data set groups is that different free-space parameters can be specified for both the volatile and nonvolatile portions of the database.

Utility programs such as DB tools and DB analyzer can be run to provide a health check of the database. The programs help in determining whether a reorganization is currently necessary and in predicting whether a reorganization may be required in the near future. This approach requires that periodic health checks be made and that the current database condition be compared with prior health checks. A program of continuous monitoring is the best gauge for knowing if and when a reorganization is required.
12.2.5 Program Isolation

When several applications need to access the same small set of database data, considerable IWAIT time (PI=) waiting for reserved (locked) resources is likely. Avoid single-record databases and databases with one record for control totals. Some examples are table databases, records for accounting, statistics reporting, security checking, and rate information. Update activity against this type of database causes transactions to single thread through the database.

A possible alternative to redesigning databases of this type, which are primarily read only, but where there are some updates, is to use the GOT or GON processing option for the read programs.

12.2.5.1 Program Isolation Wait

The duration of PI IWAITS depends on the frequency of sync point processing. Programs that update segments and have a long program elapsed time per transaction tie up resources that are required by other transactions.

If a program takes too many locks (most typically a poorly written BMP), it impacts other concurrent programs by locking data they want. Potentially more serious, it can also fill the PI ENQ/DEQ pool, having forced it to expand to its maximum size. Usually, other unrelated programs fail first, with U775 abends, when they try to take extra locks. Then the culprit fails and all of its many updates have to be backed out, again affecting the performance of the whole system.

This scenario can be avoided by judicious use of the LOCKMAX parameter. The preferred place for this is in the PSB. It is not meant to be a fine tuning parameter. Set it at a few times more than the number of locks you ever expect the program to need. For an MPP, LOCKMAX=2 (allowing up to 2000 locks) seems more than adequate. A BMP requires a higher value, perhaps LOCKMAX=15, but this must be assessed for each individual installation.

12.2.5.2 Minimizing Contention

The basic rules are:

- Specify only the PROCOPT that is actually needed. Do not specify PROCOPT=A (or R, I, or D) if the data is read only.
- If a database is rarely updated, use two PCBs: one for reading, using PROCOPT=G, and the other for the exceptional updates, using an update PROCOPT.

12.2.6 Variable-Length Segments and Compression

In HISAM, if the segment length is increased, the logical record is rewritten in its original location, and the segments that are physically adjacent to it are displaced and may be written to the overflow data set, which degrades performance. A segment with a shorter length causes shifting of segments within the logical record, which probably has no adverse effect on performance. In HDAM or HIDAM, a splitting of the prefix and data may occur when a variable-length segment is replaced with a longer segment. If the data and prefix are placed in separate blocks, the result is two I/Os to retrieve the segment.

---

20 It can also be specified in the dependent region execution parameters, but that is intended as an override in exceptional circumstances.
The minimum segment length specified in the DBD is the minimum space IMS uses on the data set. If a database is to be loaded, and it is known that a certain set of segment types will lengthen when subsequent processing begins, then specify a minimum length in the DBD to reflect the expected length of these segments rather than the length that is actually loaded. After each loaded segment, IMS leaves free space up to the minimum segment length, and this free space is available for the segment to expand into.

Segment compression exploits the variable-length segment facility. The hardware data compression (HDC) facility available on modern MVS systems makes this a very attractive option for reducing the size of DL/I databases, for fitting more segments into a block or CI, for fitting more segments into the buffer pool, and ultimately for reducing the number of I/Os. The cost of compression, to be offset against the I/O reductions, is the CPU cost of compressing and expanding segments. The MVS HDC facility is optimized for data access (segment expansion), where the cost is very modest and unlikely to be significant for many online transactions. The cost is more likely to be apparent for batch jobs that access large numbers of segments. In either case, however, it is good to be aware of a factor that can affect the cost; namely, the type of segment compression.

Two types of segment compression exist:

- Data compression
- Key compression

If you are using key compression and use a key field or a search field as qualification in your SSA, every segment that is a candidate for selection must be expanded in order to examine it. Even if you are using data compression but use a search field (not a key field) qualification in your SSA, then every segment that is a candidate must be expanded in order to examine it. However, if you are using data compression and use only the key field as qualification in the SSA, only the final segment selected requires expansion.

### 12.2.7 Data Set Groups

Data set groups are useful when segments in a database could benefit from:

- Using different block sizes
- Having different processing requirements
- Requiring separate free-space parameters, especially if the volatility of a certain segment is not typical of the others in the database

Separate segments that have widely varying lengths and that are frequently inserted into and deleted from different data set groups pose a particular problem. The reason is that the bit map in the HD organizations only indicates whether enough room is available to insert the longest segment in the data set. Any free space in the block gets used if this block is the most desirable block for an insert. However, the block is ignored by the HD space search algorithm, even if it has more than enough space for the current insert. This wasted space leads to a potential for more I/Os.

A word of caution in the use of multiple data set groups: After initial load, when the first occurrence of a segment type is inserted into the secondary data set group, there is no favored block. The first occurrence of this segment type is added to the
end of the data set group, so additional I/Os occur if the segments are processed sequentially. Insertion of these segments also results in an additional I/O, when the parent's physical child forward (PCF) pointer requires updating and the parent resides in a different data set group. Retrieval of a segment in a secondary data set group using the HD access methods results in at least two I/Os: one to the first data set group and one to the second data set group.

Implementing data set groups is one way of getting around the VSAM and OSAM data set size limit of 4 GB. In so doing, the additional I/Os must be accounted for as described.

12.2.8 Logical Relationships and Secondary Indexing

Use secondary indexes and logical relationships when such functions are required. However, overuse and misuse can lead to excessive database calls and database I/Os. These functions, if implemented properly, result in less system overhead than trying to achieve the same capability through application programming. When excessive I/Os per call are generated through the use of logical relationships or secondary indexing, use the DL/I call summary report to analyze the application program to see whether the resulting I/O profile can be improved.

The improper placement of real logical children and the choice of logical pointer options may cause additional database I/Os. We recommend that the real logical child reside on the side of the longest twin chain or on the path that is most frequently accessed. Avoid the sequencing of the virtual logical child segment.

If both sides of the logical relationship result in long physical twin chains, the frequency of access is the same from both sides of the relationship. If most access is for the retrieval of the logical child and only the concatenated key of the parent segment, consider a physically paired relationship and the use of a symbolic key. In these particular instances, by not accessing the other side of the relationship, database I/Os can be saved. Physically paired relationships also have the advantage that you can reorganize each database in the relationship at different times without affecting the other, as long as symbolic pointers have been used.

Secondary indexing is successfully implemented in many situations to reduce I/O profiles by eliminating sequential scans of databases. Note, however, that secondary indexes are ideally created from nonvolatile source segments and used for direct retrieval, using an alternate key, rather than for sequential retrieval in an alternate sequence. A good rule of thumb is to use the secondary index only when retrieving less than 10% of the target segments.

Processing the secondary index as a separate database can reduce the I/Os required to retrieve the target segment from the primary database. In this scenario, the duplicate data fields can be used to contain the nonkey data that is needed for the database calls. If the source and target segments are different, and all that is needed from the target segment is the key, consider using the /CK field in the index source segment. If you do so, the secondary index can be processed as a separate database so that the concatenated key of the target segment is retrieved with the index source segment. This saves the I/Os that would be needed to retrieve the target segment from the prime database.

Avoid secondary indexes that point into a HISAM database. A secondary index into a HISAM database must use symbolic pointers. In order to retrieve the target segment, IMS must locate the root segment through the index using the symbolic
key and then traverse top down through the hierarchy until the target segment is located.

Except in the cases mentioned above, use direct RBA pointers for the index source segment. IMS can then go directly to the segment instead of making a call from the root down to the target segment.

If the key field of a secondary index is not unique, an additional data set must be created. This ESDS is used to store segments that contain duplicate keys. In addition, an overflow chain must be created so that the duplicate keys can be located. Two methods may be used to make the secondary index unique:

- The first method is to specify a subsequence field in the DBD. This creates unique keys and the subsequence field is used to store the keys in ascending order. Application program SSAs do not change if the subsequence field is coded because subsequence fields are not seen by the program unless the secondary index is processed as a separate database.

- The second method (a subset of the first) is to specify a system-related field (/SX, which is a generated subsequence field) for HD databases. This causes the RBA of the source segment to be appended to the index pointer segment, creating unique keys. Application programs, again, use the key field in their SSAs only.

Another consideration is to use sparse indexing when only a subset of the records is accessed through the secondary index. Sparse indexing can be accomplished by coding the NULLVAL parameter on the XDFLD statement in the DBD or by employing a user-written exit routine. This eliminates the overhead associated with index maintenance for unused index pointer segments.

12.3 Application Program Design

The primary objective of application program design for performance is to keep the transaction profile at a minimum for the unit of work to be performed. Review application programs that use the most system resources first (see 4.2.7.1, “Application Programs” on page 105), because changes to these programs have the greatest impact on overall system performance. Use the DL/I call summary report (see Figure 36 on page 104) and program I/O report (see Figure 38 on page 107) evaluate the DL/I call and I/O activity for each program.

Knowledge of the application programs is helpful in making program evaluations. The DL/I call image trace program, which allows you to trace and recover all full-function DL/I calls issued by an application program, may be used to verify that the call patterns actually generated in program execution do match those predicted in the program design. The Batch Terminal Simulator II (BTS II), program product, can also be used to analyze the DL/I call patterns and SSAs used by each program. Chapter 8 of the IMS/ESA V5 Application Programming Design Guide (SC26-8016), describes the DL/I call trace program.
12.3.1 Number of DL/I Calls

The number of DL/I calls can be reduced in the following ways:

- Path calls can be used to retrieve more than one database segment per call. Avoid programs that issue multiple calls for the same segment during a single execution.
- Convert output messages from multiple ISRT calls to a single ISRT call per message, screen, or page, using MFS.
- Attempt to eliminate redundant calls and nonproductive calls (those resulting in GE, GB, or II status codes), except where the status codes represent exceptional conditions. For example, do not follow a GU call with an ISRT if a GE status code is returned from the GU call. Issue the ISRT call and check whether a II status code is returned from the ISRT call.
- Replace GN calls that retrieve large numbers of unneeded segments with fully qualified GU calls.
- Use single-segment input message specification where possible to eliminate the nonproductive GN call to the message queue. Single-segment messages require less overhead for the application program, message queue manager, message format services, and the communications analyzer.

12.3.2 Message Priming

IMS, as part of scheduling, retrieves the input message and makes it available in the dependent-region interregion communication area. This is called message priming. If the first DL/I call is not the IOPCB GU, then the primed message is lost, and will be retrieved from the queue again when the GU is eventually issued.

If the application does not issue a GU as its first call, the most likely reason is that it uses the INIT call. It is preferable to issue the GU first, and then the INIT call.

12.3.3 Number of I/Os per Transaction

The number of I/Os per transaction can be reduced in the following ways:

- Avoid batch-type processing in an online environment. Batch-type processing includes sequential scans of databases and report writing to online terminals. When this type of activity is necessary, schedule it for hours of nonpeak volume if possible.
- Do not use the ROLL call as common practice. The ROLL call is treated by IMS as an application program abend. Dynamic backout can more than double the elapsed time of the transaction, and the amount of net work accomplished for the transaction scheduled is zero.

After the transaction that issues the ROLL call is backed out, IMS issues a U777 abend to terminate the program controller for that dependent region. When the dependent region program controller is reattached, IMS again preloads all the nonreentrant preload programs, and the dynamic BLDL list is reinitialized. This can delay the availability of the dependent region for seconds and even minutes if the preload list is large.

Use the ROLB call instead of the ROLL call. ROLB backs out the transaction and returns control to the application program, which can then decide what to do. The ROLB call does not terminate the program controller.
The ROLL or ROLB call, if it has to be used, may be more efficient than scheduling another program to logically back out a transaction.

- Use common message formats to reduce storage size and I/O activity in the MFS format pool.
- Avoid large conversational SPAs. Define the maximum SPA size for active transactions to be smaller than the LGMSG message queue LRECL length (about 2 KB) if possible. SPAs are passed through the message queues for recovery, so large SPAs cause additional message queue I/O and log I/O activity.

12.3.4 Multiple Transaction Codes for the Same Program

Multiple transaction codes per application program cause additional scheduling overhead and interference, because scheduling is by transaction rather than by application program. A scheduled program processes only one transaction code per scheduling. Alternatives are transaction subcodes in the message text or separate programs for each transaction. Specification of parallel scheduling alleviates the problem a little by allowing the program to schedule in another region with a different transaction code. Refer to 10.2.6, “Parallel Scheduling” on page 212 for more detailed information on parallel scheduling in IMS.

12.3.5 Program Structure

Application programs can be structured for a paging environment by putting common mainline code adjacent to, and exceptional condition routines outside of, the mainline code in order to keep the size of the working set small. This technique is more effective for high-volume transaction programs that are also candidates for program preload.

12.3.6 MODE=SNGL and MODE=MULT Parameter

Specify the transaction parameter MODE=SNGL for all transactions to avoid locking resources longer than necessary. SNGL specifies that the altered database records are to be written back to the database upon each request for a new message (GU call to the IOPCB). If MODE=MULT (the default) is specified, the database buffers are written at program termination time and therefore tend to tie up more database blocks and buffers, and for longer than necessary.

If MODE=MULT is specified, a delay in response to the output destination is also experienced. This is because IMS sends the output message at program termination, instead of at sync point (GU to the IOPCB). The exception to this is a message to an EXPRESS PCB, which is always sent immediately.

12.3.7 Deadlocks

Deadlocks sometimes arise due to poor application or database design, and the occasional deadlock can happen just by chance. Investigate anything more than a small number per day.

If a deadlock occurs, one of the deadlock programs is pseudo-abended (U777), and the event is logged. IMSPARS reports it in the management exception report, in the section headed “Error Conditions” (see Figure 93 on page 290).
This report tells you only which transaction and program abended. The IMS monitor can provide more useful information, however. Deadlock events are reported by the IMS monitor in the “Reports” report (Figure 94).

These events are U777 abends, and the report shows the winning and losing PSBs.

Because the IMS monitor is run for only short periods, it is quite likely that none of the deadlocks will be reported by the monitor. In this case, it is necessary to return to the IMS log and use the IMS DFSERA10 file select and formatting print utility, specifying DFSERA30 as an exit. This produces a deadlock report for each event. This report is very detailed and a full explanation can be found in Chapter 11 of the IMS/ESA Version 5 Utilities Reference: System (SC26-8035).

It is important to analyze deadlocks because, when the deadlock is detected, IMS backs out the transaction that lost, terminates the dependent region program controller task, and reattaches the program controller task. This sequence of events causes the nonreentrant preloaded programs to be deleted and reloaded and causes the dynamic BLDL list that IMS maintains to be reinitialized.

Furthermore, deadlocks can impact other transactions in that extra data has to be written to the log (type X ‘67FF’ records) and the log has to be read to perform the backout.

If the occurrence of deadlocks becomes high, a possible strategy is to examine the class scheduling structure of the transactions involved and to attempt to funnel conflicting transactions through a single dependent region dedicated to problem
transactions. This is a short-term solution. The long-term solution is to modify the applications to issue DL/I calls in a way that does not cause deadlocking.

### 12.4 Batch Message Processing

The IMS environment provides the capability to process batch style jobs against databases that are being referenced by online programs. The IMS Batch Message Processing (BMP) Region runs under the control of IMS as though it were an online transaction processing region, using the IMS system log.

With increased pressure to have systems available 24 hours a day, 7 days a week, the opportunity to stop IMS from accessing the databases in order to allow batch jobs to complete (the old batch window) is all but gone. BMP processing has become the only way installations can manage to get large batch schedules, such as at month end, to complete in time with full data integrity.

BMPs have a further major advantage over pure batch jobs that run outside of the control of IMS. BMPs have the ability to access DEDB databases, regarded as the standard for high-performance environments.

As you might expect, the very difference in nature from online transaction processing makes BMPs candidates for performance problems. The system must be carefully designed to allow both to run concurrently at optimum efficiency. If this is not done, BMPs could monopolize system resources, causing delays and unacceptable online response times. Online transaction regions cannot be given a higher priority in terms of IMS resources, and the two compete on an equal footing.

A critical issue to analyze is the frequency of checkpoints the BMP takes while executing. If a checkpoint is issued too frequently, the CPU required to process the checkpoint increases. If the extended checkpoint option is used, large amounts of data can be written to the log, delaying online transactions and Fast Path sync points that are attempting to write to the log at the same time. Each checkpoint also causes a message to be sent to the IMS master terminal. We strongly recommend that this be suppressed. Two techniques that accomplish this are discussed in 12.4.1.3, “Checkpoint Message Suppression” on page 292.

If, on the other hand, the time between checkpoints is too long, it is possible that the BMP is holding enqueues on database records that online transactions require. The online transactions are then put in a PI IWAIT, waiting for the records to be released. This could take a long time, depending on what processing the BMP was doing.

### 12.4.1 Checkpoint Calls

A checkpoint call indicates to IMS that a program has reached a commit point. This call also establishes a point in the program from which the program can be restarted. There are two kinds of checkpoint calls a program can issue:

- Basic CHKP for MPPs, BMPs, or IMS batch (rarely used) programs
- Symbolic or extended CHKP for BMPs or IMS batch programs only, used in conjunction with the extended restart call, XRST

These enable you to save as many as seven program data areas and restore them when the extended restart (XRST) call is issued upon restarting after an abnormal termination.
This section does not go into the detail of using these calls but reviews some of the performance implications of using or not using the symbolic checkpoint call in BMPs. For a more detailed explanation of the use of these calls, refer to IMS/ESA 5.1 Application Programming: Design Guide (SC26-8016).

12.4.1.1 Checkpoint Action Taken
When the BMP issues a checkpoint call, IMS does the following:

- Commits database updates.
- Logs all modified database records before they are written to the DASD device.
- Frees all locks on the data.
- Writes the user areas (up to seven specified in the CHKP call) to the IMS log.
- Checkpoints GSAM files.
- Writes a log record containing the checkpoint identification given in the call.
- Writes a message containing the checkpoint identification to the IMS master terminal and MVS console (JES job log).
- Returns the next input message to the program I/O area if the program processes input messages.

12.4.1.2 Reasons for Checkpointing
There are two reasons why checkpoint calls are important in BMP programs, especially if the BMP retrieves and updates many database records, or even if it performs very few updates but just runs for an extended period:

- The database records are locked out from use by other application programs for long periods of time. The exception to this is a BMP that has a processing option of GO, GON, GOT, or EXCLUSIVE specified in the PCB (no locking is available with these PROCOPTs).

Once the checkpoint call is completed, those segments that were held are released for use by the other application programs.

- The amount of virtual storage required to hold the ENQ/DEQ control blocks may exceed the size specified in the PIMAX parameter in the control region execution parameter. This is discussed in 12.2.5, “Program Isolation” on page 284.

12.4.1.3 Checkpoint Message Suppression
When a BMP (or a batch program) issues a CHKP call, a message is produced to be sent to the master terminal operator (MTO) and the MVS console. This can be a problem if applications are checkpointing very frequently, placing a significant burden on the message queues. The message content is not actually needed for BMPs, because BMP restart is always done with CKPTID=LAST. Consequently, it is possible to suppress the issuing of the checkpoint messages. There are two ways of achieving this:

- Use the dependent region execution parameter

```plaintext
CKPTID='NOMSG'
```

If a BMP restart is necessary, IMS remembers that no messages are required when the user specifies CKPTID=LAST.
• Use the systemwide option in DFSVSMxx

```
OPTION ISSUE681=NONE
```

Other values are available for ISSUE681=

- ALL: Do not suppress any checkpoint messages.
- nn: Issue no more than nn checkpoint messages per second per program. If the checkpoint frequency is greater than this for any program, notify the MTO (because the program is probably looping).

### 12.4.1.4 Checkpoint Frequency for BMPs

The question, “How frequently should I take a checkpoint in my BMP?” is often asked. No one answer to this question exists. The answer depends on the environment in which the BMP is executing. What may be a good frequency in one installation or IMS system may not be good in another.

The frequency of checkpoint calls depends on how much work a BMP does. The minimum frequency between checkpoint calls should be one unit of work. A unit of work in this environment is the work that an application program must complete (before a checkpoint is taken) in order to avoid the loss of position within the database record that is currently being processed. Database position is lost at CHKP time. It is wasteful to have to reposition within the same database records after checkpoint, especially if repositioning down twin chains is involved. With DEDB processing, repositioning is almost guaranteed to force CI contention of a BMP with itself, as it tries to access a CI that is waiting to be written out by an OTHREAD.

In practice, most BMPs do more than one unit of work per checkpoint. Checkpoints are frequently taken, for example, after processing 10 records read from a sequential input file or after processing 25 database records being read sequentially. We recommend that the time period between checkpoints be in the range of one to a few seconds. In fact, for DEDB sequential updating BMPs (with PROCOPT=P or H), it is quite common to get four or five checkpoints per second per BMP.

Sometimes BMPs are run with checkpoint intervals measured in tens of seconds or in minutes. If this is practical and causes no contentions, then clearly it is satisfactory.

Ease of changing the checkpointing frequency for a program is an important aspect to consider. Various ways are suggested:

- Supply a value to the BMP with the dependent region execution parameter APARM=. This is picked up by the BMP by issuing an INQ DL/I call, with the subfunction ENVIRON.
- Pass a value in SYSIN data.
- Set up a checkpoint control record in a database (or OS file) that is read by the BMP.

The frequency of checkpoint calls can be specified in terms of the number of updates the application program does, a time interval, or both. A time interval, for example, can be specified and, once the time expires, the BMP finishes the current unit of work and then takes the checkpoint.
Minimizing Lock Duration for Reads: A problem that is sometimes encountered is where a BMP reads a record from one PCB and then performs the rest of its processing on other PCBs up to the next checkpoint. The first read could cause unnecessary contention delays for other programs. The problem can be avoided in the following ways:

- Use PROCOPT with caution. For example, do not use PROCOPT=A if the data is read and never written.
- Use two PCBs if a database is rarely updated, and only in exception cases: one for the reads and one for the exceptional updates.
- Use PROCOPT=GOx for read-only PCBs if no danger of GG status codes exists.
- Release the lock after reading a record by issuing a GU call for a nonexistent root (key=X'FF', for example).

12.4.1.5 Checkpoint Data Written to the IMS Log
When the checkpoint call is issued, IMS writes up to seven user areas, as specified in the CHKP call, to the IMS log. The length of time it takes to complete the BMP checkpoint call depends on the amount of data being written to the log. The more data written, the longer the checkpoint call takes.

Analyze the size of the user data being checkpointed to determine whether all of the data is required for restart. If all of the data is not required, reduce the content of the data set to that required to restart the BMP. In other words, define the application fields that genuinely need checkpointing and recovering at XRST time in the minimum number of pieces of contiguous working storage. Specify only these pieces of storage on the CHKP calls.

This prevents frequent BMP checkpoints from monopolizing the log and possibly impacting other work in the system.

12.4.2 BMP Dispatching Priority
Assign the dispatching priority of the BMP above that of the MVS batch and that of the online message regions. It is difficult to recommend a priority in relation to TSO. The BMPs hold IMS latches just as the online regions do, and, if the BMPs cannot be dispatched, the IMS control regions or the dependent regions may have to wait until the BMP is redispached by MVS and releases the latch.

This could be microseconds or many hundreds of milliseconds, depending on the load on the MVS system. Therefore, your installation needs to be told whether the BMPs are more important than at least second and third period TSO. It is practical to place the BMP dispatching priority below that of the TSO first period but above that of the second and third periods.

12.4.3 Planning for BMP Execution
Carefully consider the total number of BMPs that are scheduled to execute concurrently within the online system. The size of each of the different buffer pools needs to be adjusted accordingly. These pools include the PSB, PSBW, and DMB (if all DMBs are not resident).

If a database is often scanned sequentially by BMPs, then that database is a prime candidate for its own database buffer subpool.
12.5 PROCOPT=GOT with DBRC SHARECTL

We strongly recommend that you use DBRC SHARECTL and register all databases in a production environment. However, be aware that IMS performance degradation can occur when using PROCOPT=GOT and DBRC SHARECTL. The degradation applies to all users and may well apply to databases other than the one being accessed with PROCOPT=GOT. This applies to IMS DB/TM, DBCTL, CICS, and batch sharing environments.

If an invalid pointer is detected when using PROCOPT=GON or GOT, then processing is retired. With PROCOPT=GON, retry processing simply retries the call and returns a GG status code if the problem is still present. The retry processing for PROCOPT=GOT consists of two actions:

- If a locking environment exists, a test enqueue on the database record containing the segment being retrieved is requested. (A locking environment exists for an IMS DB/TM, IMS DBCTL, or CICS system and also for IMS batch jobs if block-level data sharing is being used.) The test enqueue causes call processing to wait until conflicting holders of locks on the database record release their locks.

- The last block read for the call is reread. This is an attempt to get the latest image of the block.

In many cases, this retry succeeds in correcting the problem that produced the apparent invalid pointer condition. However, it may also cause serious performance degradation in the IMS subsystem. The degradation may affect databases for which PROCOPT=GOT is not used. As well as causing a reread of the last block (Step 2 above), buffer invalidation takes place. This causes all the buffers for all databases that are registered at share levels 1, 2, or 3 to be invalidated. The lookahead process will not find any blocks in the pool for these databases until they are reread from DASD. Hiperspace buffers and OSAM sequential buffering buffers are invalidated, just as are other buffers.

If the use of PROCOPT=GOT causes unacceptable performance degradation, consider changing PROCOPT=GOT to PROCOPT=GON.

We strongly recommend you use PROCOPT=GOT only if contention with updaters, though possible, is highly unlikely.

12.6 Generalized Sequential Access Method

GSAM is commonly used in both stand-alone batch and BMPs for reading and writing sequential files. Its key advantage is that IMS provides checkpoint and restart support for the GSAM files when the program uses CHKP and XRST calls.

Although GSAM uses standard MVS sequential file support, two aspects can have a significant impact on performance: the choice of PROCOPT and the unnecessary use of RSA.
12.6.1 GSAM PROCOPT
Using multiple buffers requires that the S processing option be included. The recommended PROCOPTs are:

<table>
<thead>
<tr>
<th>Usage</th>
<th>PROCOPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>PROCOPT=GS</td>
</tr>
<tr>
<td>Write</td>
<td>PROCOPT=LS</td>
</tr>
</tbody>
</table>

12.6.2 Unnecessary Use of RSA
The standard DL/I calls for GSAM are GN (to read a record) and ISRT (to write a record). Both calls have an optional parameter, RSA.

Do not use the RSA parameter unless it is actually required. To obtain the RSA value, IMS issues a NOTE macro, which uses a very significant amount of CPU, and in some cases can cause additional I/Os.

12.7 ROLL and ROLB Calls
When an application program determines that some of its processing is invalid, two IMS calls make it possible for the program to remove the effects of the incorrect processing. These are the rollback calls, ROLL and ROLB. When either of these calls is issued, the following actions take place:

- IMS backs out the database updates that the program has made since the program's most recent commit point.
- IMS cancels the output messages that the program has created since the program's most recent commit point, except for express output PCBs.

The main difference between the two calls is that:

- The ROLB call returns control to the application program after backing out updates and canceling output messages and is valid only in single-mode programs (MODE=SNGL in TRANSACT macro).
- The ROLL call terminates the program with a pseudo abend code of U0778 and is valid in both single-mode and multiple-mode programs (MODE=SNGL or MULT in TRANSACT macro).

In the current IMS environment, we recommend that all IMS transaction programs be single-mode, unless there is a well-defined reason for them not to be.

This book discusses only the performance aspects of these calls. For more detailed information on these calls, refer to IMS/ESA V5 Application Programming: Design Guide (SC26-3066).
12.7.1 Use of the ROLB Call

The use of the ROLB call is recommended in an IMS online environment for performance reasons. The ROLB call, as previously described, returns control to the application program after IMS backs out the database changes. At this time, the application program can send a message back to the inputting terminal explaining why a normal response was not sent (or anything else the program might want to do). It then goes to normal termination. There is no noticeable effect on IMS resources except the resources required to back out the changes already made. The time and resources can be minimal if the program did very little work.

12.7.2 Use of the ROLL Call

We do not recommend using the ROLL call in an IMS online system because of the effect it has on the performance of the message region in which it executes. The IMS processing needed to back out the database changes is the same as in the description of the ROLB call. The difference in the two calls is the processing that is done after IMS finishes backing out the changes. Unlike ROLB, the ROLL call terminates the application program with a U0778 pseudo abend. This abend also terminates the program control task within the MVS region. The IMS region control task then reattaches the program control task and, at this time, the following occurs:

- All the nonreentrant preloaded programs are preloaded again.
- The dynamic BLDL list storage is reinitialized.

The time that the region is unavailable for use by other transactions that can be scheduled in the region depends on the number of programs that are preloaded in the region. This time can be as short as a few hundred milliseconds or it can be as long as minutes if the preload list is long.

Therefore, to perform application rollback, the ROLB call is strongly recommended.

12.8 Advanced Program to Program Communication

APPC is a major evolutionary change to MVS for applications that need to connect and communicate across the enterprise.

12.8.1 APPC/IMS

The APPC/IMS feature of IMS enables APPC conversations to be established between IMS applications and partner applications. Either partner can allocate the conversation.

The APPC architecture, based on LU 6.2 protocols, has some significant differences from the traditional IMS messaging architecture. Consequently, several performance implications need to be understood to ensure that performance problems are avoided.

APPC/IMS uses the services of APPC/MVS.
12.8.2 APPC/MVS

APPC/MVS, a VTAM application that extends APPC function to programs in an MVS system, providing LU 6.2 capability to programs running in MVS. APPC/MVS is implemented as a started task, running in a separate MVS address space. It has also implemented its own scheduler address space, ASCH; when used, ASCH controls a pool of address spaces for MVS transaction scheduling.

The APPC address space receives and processes inbound requests as well as outbound requests. Inbound requests are validated by APPC/MVS and then passed to an appropriate address space for processing. (Processing may consist of transaction scheduling or use of MVS server facilities.)

The APPC/MVS transaction scheduler maintains pools of address spaces called APPC/MVS transaction scheduler initiators (ASCHINTs), which are similar to batch initiators. These address spaces are managed by the APPC/MVS transaction scheduler.

12.8.3 Role of APPC/MVS for APPC/IMS

With IMS, the APPC/MVS scheduling functions are not used (although the ASCH address space still needs to exist). Instead, MVS posts an IMS scheduling exit whenever it receives an allocation request for an IMS application.

APPC/MVS provides the APPC application programming interface (API) to IMS. In fact, two APIs are available. IMS uses the APPC/MVS API (for example, on behalf of the application when executing a traditional IMS DC application). APPC application programs, running in IMS dependent regions, can use either the common programming interface for communications (CPIC) or the MVS API.

APPC/MVS provides all the message buffering services, and is the interface between IMS and APPC/VTAM.

When an LU 6.2 application allocates a conversation, the ALLOC command specifies the target LU (IMS) and the target transaction process name (TPN). The TPN can be up to 64 characters long. MVS has a TP_PROFILE data set from which it reads a profile for each of the LUs it is servicing. An IMS profile can contain an entry for all, some, or none of the transactions. It is used, if necessary, to map TPN into a standard IMS eight-character name. For an existing transaction, this name is simply the transaction code. However, APPC/IM also supports applications using an APPC API. In this case, the profile maps the TPN into a program name and an IMS scheduling class. These two types of transactions are discussed in the following sections on implicit and explicit support.

The APPC/IMS facility is described fully in Client/Server Computing with IMS Using APPC (GG24-3981).

12.8.4 APPC/IMS Application Processing

APPC/IMS is a part of the IMS Transaction Manager (TM) and supports two distinct modes of application processing, implicit and explicit.

Implicit support enables IMS application programs to be invoked by, or to allocate, APPC conversations without being sensitive to the LU 6.2 protocols. Normal DL/I calls are issued by the application program, and therefore existing programs can operate unchanged with LU 6.2 devices. All the APPC/MVS calls used to manage
the conversation are issued by IMS. APPC/IMS receives the input message and puts it on the IMS message queue (or into the EMH buffer). Similarly, replies, where necessary, are sent via the message queues.

Using the implicit API allows most existing applications to continue to function unchanged. Existing applications that are sensitive to a terminal's hardware characteristics, such as cursor position, may have to be changed to communicate with LU 6.2 devices. Applications that are sensitive to the MFS MOD name or the LTERM name can execute unchanged, but require the use of an APPC/IMS user exit and probably cooperation from the remote partner application.

Explicit support requires the IMS application to issue CPIC (or MVS API) calls such as CM_ACCEPT, CM_RECEIVE and CM_SEND rather than issuing IOPCB DL/I calls. For native APPC communications, IMS does not look at or log the messages. Similarly, no APPC-related events are written to the IMS monitor.

Explicit mode transactions are scheduled by the IMS scheduler and, therefore, generate scheduling records on the IMS monitor. Clearly no IOPCB calls are made to the message queue, but the APSB counts as the first DL/I call as reported by the monitor in its schedule-end-to-first-DL/I-call values.

For explicit mode transactions no, X'08' or X'07' scheduling and termination log records are produced. However, APPC equivalents are written, namely X'0A08' and X'0A07' log records.

12.8.5 APPC/IMS Performance Monitoring

Implicit mode transactions are standard IMS transactions and consequently produce all the standard log and monitor records.

Modified applications do not create any log or monitor data for the spawned APPC conversations.

Explicit mode transactions are scheduled by the IMS scheduler and, therefore, generate scheduling records on the IMS monitor. Clearly no IOPCB calls are made to the message queue, but the APSB counts as the first DL/I call as reported by the monitor in its schedule-end-to-first-DL/I-call values.
12.8.6 APPC Application Design Considerations

Some APPC conversational characteristics are described in the following to provide an understanding of the performance implications. Aspects of application and data integrity are not discussed.

12.8.6.1 APPC Characteristics

A conversation takes place between two programs somewhere in the network. They may be on the same logical unit (LU) or widely separated; it does not matter to the programs, although it may well influence performance.

The initiator of the conversation specifies certain attributes for the conversation on the allocation request. The partner can accept or reject the conversation, based on the attributes and security checks.

The conversation initiator identifies the partner LU and transaction program name (TPN), and sends them in the SNA FMH5 (Attach FMH). Because the TPN can be 64 characters long, the FMH5 can be substantially larger than usually seen in the network. Suitable allowances must be made in network capacity estimates if all the fields in the FMH5 are near maximum length. (The maximum length could be around 150 bytes, whereas the data might be only 20 bytes.)

A significant architectural difference between APPC and traditional IMS messaging affects how reply messages are sent. The standard IMS technique is for IMS to take responsibility for application output messages created with ISRT calls. It secures them on the message queues and ensures they are recoverable by logging them. At sync point time, it is unnecessary for the application to wait on message delivery, as message integrity is the role of IMS.

In APPC architecture, it is the application that is responsible for message integrity, not the system. Consequently, sends are required to complete successfully before sync point processing begins. If sync point fails, the partner is told and can choose to discard any replies it previously received. The performance implications of this depend on the conversation attribute sync_level, which is discussed below.

12.8.6.2 Conversation Attributes

The conversation type may be MAPPED or BASIC. This has no perceived effect on performance.

Sync_level - determines options that may be requested by the partner applications when they communicate. Three values are defined in the LU 6.2 architecture, NONE, CONFIRM, and SYNCPOINT. IMS Version 5 supports the first two options. A sync_level of NONE means that no acknowledgment can be requested when an application sends data to its partner.

A sync_level of CONFIRM allows an application, when it issues a SEND, to request confirmation that data has been received by the partner. The partner provides the confirmation by replying with CONFIRMED.

The implications of using sync_level=CONFIRM are important, for two reasons:

- Additional network traffic is incurred for the confirmation.
- The application requesting the confirmation has to wait for the acknowledgment, and this includes time spent transmitting the data to the
partner and transmitting CONFIRMED back to the waiting application. This time can be significantly greater than the real transaction processing time.

There are two conversational styles:

- **Asynchronous**, where the initiator allocates a conversation, sends a message, and deallocates the conversation without waiting for a reply. The verbs used are ALLOC, SEND, and DEALLOC. A new similar conversation is allocated by the partner for the reply. The classic message pair is therefore two simplex conversations.

- **Synchronous**, where the initiator waits in the conversational state for a reply to be received. The verbs issued are ALLOC, SEND, RECEIVE, and DEALLOC. Here, the conversation is half duplex, in that both partners can send and receive, but only one can “speak” at a time.

**12.8.6.3 IMS Transaction Options Supported**

Implicit support allows all types of IMS transactions:

- Response mode (must use a synchronous conversation)
- Nonresponse mode (synchronous or asynchronous conversations permitted)
- IMS conversational (APPC conversation maps onto the IMS conversation)
- Fast Path expedited message handler

**12.8.6.4 Using IMS Explicit Mode**

Application design can be as simple or complex as required, but bear in mind the implications for the whole system. An explicit-mode conversation can involve many interactions between the remote application and the IMS application, perhaps continuing for minutes or even hours. If this is the case, make the message-processing regions distinct from those used for regular IMS transactions!

Class scheduling is strongly recommended. Keep the class values to the lower end to avoid the system reporting on null classes. The default class is 1 in all cases.

Region occupancy issues are self-evident in this environment. However, it must never be overlooked that update locks taken during the conversation are retained until commit point (using the SRRCMIT call). Furthermore, it is not just update locks that need to be considered. As for BMPs, a read lock, taken when a get call is issued against a PCB, is retained until database position moves off that record with that PCB. A long-term read lock prevents updaters from accessing that database record.

Even if the explicit application is a simple transaction program, the region-occupancy implications of using sync_level=CONFIRM must also be considered. As explained above, the time-in-system using CONFIRM is likely to be many times longer than when using a sync_level of NONE.

The consequence of very high region occupancy is that far more dependent regions are needed. IMS allows up to 999 regions, but note that every region needs storage resources. So consideration must be given to the sizes of many pools: PSB, PSBW, PI ENQ/DEQ, and database pools in particular.
12.8.6.5 Using IMS Implicit Mode

Because this appears to be traditional IMS, it is easy to overlook the performance implications of implicit mode processing.

**Asynchronous Conversations:** Asynchronous conversations must use non-response-mode IMS transactions. No particular performance issues need consideration. If the message processing program creates a reply, IMS queues it and after sync point, allocates a new conversation. IMS uses sync_level=CONFIRM for message integrity reasons. Only when the CONFIRMED eventually arrives does IMS dequeue the reply.

**Synchronous Conversations:** With sync_level=NONE, the performance implications are minor. At sync point time (when IMS application issues IOPCB GU), IMS issues the APPC/MVS SEND, and on receiving a good return code from MVS, immediately continues with sync point processing.

With sync_level=CONFIRM, major performance implications must be considered. At sync point time, IMS sends the reply requesting confirmation, and the message processing program remains dormant in the dependent region with all the associated IMS resources still allocated (PSB, locks, database buffers, and other resources).

The duration of this wait in a network environment is typically 2 or 3 seconds, compared with the normal sync point time of a fraction of a second. So, for example, if such a transaction normally runs in 0.25 second, it may well take ten times as long in the APPC environment. This probably means specification of ten times as many regions to service the transaction.

Give serious consideration as to if and when a sync_level of CONFIRM needs to be specified. With any sync_level the possibility of getting in-doubt updates exists if communication between the partners is broken (because one partner does not know if the other committed its updates). The chances of this happening, when a communications link breaks, are much higher with a sync_level of NONE, but it can also happen with CONFIRM. The APPC architecture does not include any concept of conversation recovery, so it is an application responsibility to resolve these indoubt situations, using a new conversation, when the link is restarted.

Because integrity needs to be provided by the applications regardless of the sync_level, some have questioned the value of using CONFIRM when the performance cost is so significant. On the other hand, many installations run IMS with the knowledge that in the unlikely event of an enforced cold start, they may lose messages and in particular, output messages. Some protection may be gained by using the message requeue product, but you may be willing to accept the small risk of losing messages. You may prefer to use a sync_level of CONFIRM to minimize the number of in-doubt situations, while accepting the small risk of their occurring, just as they already do in the enforced cold-start scenario.

12.8.7 Security Considerations for APPC/IMS

Security with APPC is more complex than with other LU types. With LU 2, for instance, the user typically signs on, and then enters many transactions, all of which are associated with the same user ID.

With APPC, every allocation request is considered to be a new sign on, and a user ID is passed as part of the allocation request (in the FMH5). Depending on the
remote LU 6.2 platform, the allocation may include the indicator ALREADY
VERIFIED, in which case APPC/MVS does not need to check the user's password.
Otherwise, a password must be supplied and RACF (or equivalent) is called to
verify the user.

For each allocate routine, APPC/MVS then calls RACF (or equivalent) to check that
the user ID is authorized to use the TPN. This is a mandatory check and cannot
be eliminated.

If both security checks are successful, the transaction passes to IMS. If IMS uses
transaction security, then by default, a check is made to ensure that this transaction
is authorized for use by this user ID. However, because the equivalent check has
already been made on the TPN by MVS, this may be considered unnecessary.

For APPC input messages only, this aspect of security can be controlled by using
either:

- TP_PROFILE entry for TPN, specifying RACF=xxxx

or

- The IMS command /SECURE APPC xxxx

where xxxx has the value NONE, CHECK, or FULL.

These values have the following implications:

**NONE**  No IMS transaction authorization takes place. The user's is available in
the IOPCB and passes, for example, to DB2 for PLAN authorization.

There is no accessory environment element (ACEE) in either the control
region or the dependent region.

**CHECK**  Normal IMS authorization processing occurs. MVS passes the RACF
UTOKEN to IMS, and IMS invokes RACF to build an ACEE in the
control region.

**FULL**  Normal IMS authorization processing occurs. In this case, IMS also
uses the UTOKEN to build an ACEE in the dependent region. This is
necessary if any non-IMS authorized services are required (data set
open, access to MQM, or other services).

Clearly, many more calls to RACF (or equivalent) are made in the APPC
environment. It is unacceptable if RACF I/O must be performed every time.
Fortunately, RACF (starting in RACF 1.9.2) provides a facility ACEE data in
memory, where it holds the most recently used ACEEs in a VLF data space. This
requires the following member in SYS1.PARMLIB:

```
CLASS NAME(IRRACEE)
EMAJ(ACEE)
MAXVIRT(nnn)
```

where nnn is the storage to be allocated for ACEE caching
12.8.8 Recommendations

For Implicit Mode, follow the recommendations in Table 17

<table>
<thead>
<tr>
<th>IMS Transaction Type</th>
<th>APPC Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonresponse mode transactions, inquiry only</td>
<td>Asynchronous and SYNC_LEVEL=NONE</td>
</tr>
<tr>
<td>Nonresponse mode transactions, update</td>
<td>Asynchronous and SYNC_LEVEL=CONFIRM</td>
</tr>
<tr>
<td>Response mode transactions (including Fast Path and IMS conversations)</td>
<td>Synchronous and SYNC_LEVEL=NONE</td>
</tr>
</tbody>
</table>

Where possible, redefine response mode transactions as (or define APPC variants as) nonresponse mode transactions.

For security, do the following:

- Include RACF=NONE in TP_PROFILE.
- Perform transaction security in APPC/MVS.
- Use ACEE data in memory.
Chapter 13. Database Recovery Control (DBRC) Performance

DBRC has for many years been a mandatory component of IMS DB/TM and DBCTL systems, and an option when running batch jobs that access IMS databases. It is thus a critical component of any IMS system and must be considered when looking at the performance of the system.

DBRC can be initialized at three different levels, depending on what function it is required to perform. The three levels of DBRC are:

1. Log control level. DBRC records only information about IMS system logs.

2. Recovery control level. DBRC records information about the IMS system logs, as with log control level, and in addition records information about image copies, change accumulation, database recoveries and reorganization for databases that are registered with DBRC. It also automates the generation of JCL for the IMS utilities (IC, CA, RECOV).

3. Share control level. DBRC adds support for data sharing (database or block level) to the functions described in log and recovery controls levels. It also includes protection of databases from misuse (accidental or otherwise) in both a data-sharing and a nonsharing environment.

It is strongly recommended, for all production systems, that DBRC be implemented at the share control level and that all databases be registered. This recommendation means that DBRC is subject to maximum usage, so performance considerations are of great importance.

13.1 Chapter Scope and Structure

The chapter focuses on the performance of DBRC. Implementation and use of DBRC are not covered.

The chapter is structured as follows:

- DBRC functions
- The RECON datasets
- RECON access, performance considerations
- DBRC buffers

13.2 DBRC Functions

In an environment making use of DBRC at a control level of SHARE, DBRC is required to control access to the databases from the various MVS subsystems. It is in this scenario that performance needs to be seriously considered, as DBRC remains a single-threaded process.

Figure 95 on page 306 lists the key functions in which DBRC is involved, and which could affect the system performance.
Of the above functions the following are the most interesting.

13.2.1.1 IMS SignOn
During IMS startup, the following actions are taken by IMS and DBRC:

- The DBRC address space is initialized for IMS DB/TM or DBCTL subsystems.
- Subsystem signon occurs only in a DBRC share control environment, for IMS Batch as well as for IMS DB/TM or DBCTL. A SUBSYS record is created.

The signon process is not normally a performance issue as it occurs at IMS startup. However, if a number of batch jobs are started, the signon process is repeated for each one, which can result in quite an overhead. The best advice to avoid this situation is to convert all batch jobs to BMPs.

13.2.1.2 DBRC Authorization Process
DBRC authorization process takes place:

- Once the IMS system is started.
- After a /DBR and /STA command are completed.

Full-function database authorization occurs at first schedule time and DEDB area authorization occurs at Area open time.

The RECON is referenced to check whether the database is registered with DBRC, to establish the share level, and to see if certain indicators are turned on (for example, Recovery Required). The process involves checking the access intent, specified in the database macro on the IMS sysgen (or the IMS /STA command), and the share level specified in the RECON.

The SUBSYS and DATABASE records are updated.

Database authorization is completed in a single process for all the databases associated with the PSB, and remains in effect until a /DBR command is issued against a database, or IMS is terminated.
To avoid the overhead of DBRC authorization processing and database-open processing that occurs when the database is accessed for the first time by a valid transaction, it is possible to define a PSB containing all their databases. This PSB is then scheduled using a transaction or BMP at IMS startup. It issues one call to each PCB, authorizing in the databases. For DEDBs, this can be accomplished by registering the areas in DBRC with the PREOPEN attribute.

13.2.1.3 DBRC Database Allocation
The DBRC allocation process is not to be confused with the MVS allocation process (SVC 99). In IMS 5.1, a new parameter in the /STA DATABASE command is provided to defer the MVS allocation until the database is actually opened at the first DL/I call.

The DBRC allocation process is invoked at the first update call, when the first change record is written to the log. The ALLOC record is created in the RECON, the LOGALL record is updated, and the data sharing information is returned. This process must occur individually for each database, unlike the authorization process, which allows all the databases in a PSB to be authorized at once. The only way to get the ALLOC records created ahead of the live transaction is to actually update the database. A dummy segment could be used in a BMP run in the same way as the one described under the DBRC authorization process. Whether the avoidance of the allocation processing at update time warrants a change in the structure of the database is a decision that you must make.

13.2.1.4 IMS Sign Off
The heaviest task performed by DBRC at IMS shutdown is the process of rescinding the authorization of the databases unauthorizing them (if they were not previously unauthorized by the /DBR command). Once unauthorized is completed, the sign-off process, which involves the deletion of the SUBSYS record from the RECON data set, occurs.

**Note**
If IMS is abnormally terminated, the DBRC process does not complete normally. In this scenario, emergency restart schedules this work later.

13.3 The RECON Data Sets
The availability and integrity of the RECON data sets are of paramount importance to the correct functioning of an IMS system. They are required to control logging and system restart operations and, they can also be used to control the maintenance and recovery of registered databases. DBRC serializes access to the RECON datasets to ensure that each DBRC instance can be assured the information in the RECON datasets is valid.

13.3.1 VSAM
The RECON datasets are VSAM key sequence data sets (KSDS). The *IMS/ESA V5 Installation Volume 1 (SC26-8023)* describes the VSAM parameters required to define the RECON data sets.

Do not specify the BUFFERSPACE, BUFNI, or BUFND parameters when allocating the VSAM clusters.
To avoid some unnecessary I/O, do not specify the WRITECHECK parameter in the DEFINE CLUSTER of the RECON data sets.

To avoid performance degradation, make all copies of the RECON data sets identical in their index and data CI sizes. It is recommended that the C size of the data component exceed the index component by at least 2048 bytes.

**Record Size Limitation:** The record size limitation of 32 KB for nonspanned records is often too small for IMS customers. Using spanned records removes the limitation. DBRC does not have any intrinsic limitation. However, 32 KB is the largest record size supported by the AMS REPRO and EXPORT functions when copying to sequential datasets (that is, tape). These functions are usually used to copy the RECON to a sequential file on tape. An alternative to using IDCAMS is to copy the RECON data sets using the DFSMSdss DUMP function, or just use these functions to REPRO or EXPORT the file to DASD.

The DBRC BACKUP.RECON command is subject to this restriction only if the new dataset is on tape (copying to DASD is fine).

The DBRC BACKUP.RECON command can be used to create backup copies of the RECON, but it uses AMS REPRO, and so is subject to the 32 KB limitation.

### 13.3.2 FREESPACE Specification

Make the FREESPACE parameter in the DEFINE CLUSTER of the RECON data set different for each of the three RECON data sets to avoid CI/CA splits occurring at the same place in each.

### 13.3.3 RECON Maintenance

The size of a RECON data set may also affect performance. You need to be sure that the RECON data set is being cleaned up correctly. To remove old, unwanted information from the RECON, use the DELETE.LOG INACTIVE command. Four conditions must exist before a PRILOG record is deleted by the command:

- The log must be closed (have a stop time not equal to zero). It is possible for a log not to have a stop time; for example, when a batch job abends and the log is not closed.
- The PRILOG record must be at least 24 hours old.
- The record must not be marked in error.
- The LOGALL record must have an ALLOC=0 count. This count is reset when an image copy is taken.

The DELETE.LOG INACTIVE command does not delete any interim logs. The DELETE.LOG INTERIM command must be used for this. Interim logs are created when using the log close utility DFSULTR0. They normally get removed automatically from the RECON. However, if DFSULTR0 does not complete correctly, they may be left in the RECON.

If an old log remains open in the RECON, not only are the PRILOG and SECLOG records held, but so are all the related ALLOC, reorg, and recover records. This can increase the size of the RECON substantially. The LIST.HISTORY command can be used to get an indication of whether the RECON data sets need to be cleaned up.
Do not make the GENMAX parameter, specified on the INIT.DBDS command, excessive. The parameter specifies the number of recovery generation that DBRC maintains. Each time you run an image copy, a new recovery generation is defined to DBRC. Keep the GENMAX parameter as low as possible, but within your installation's standards. It is important that all databases are regularly image copied. A single database that is infrequently image copied, and that has a large GENMAX, causes DBRC to keep data about all the logs since the oldest image copy.

13.3.4 RECON Dynamic Allocation
Dynamic allocation is recommended for the RECON data sets. Do not specify any RECON DD card in the IMS started-task or in the IMS batch job stream.

The main advantage is that all jobs automatically use the correct set of RECON data sets. A further advantage is that when an error occurs with one of the RECON data sets and DBRC automatically switches to the spare, the operator can fix the broken one. In a multiple processor environment, ensure that all the IMS systems have access to the same dynamic allocation members. That is, do not use dynamic allocation for some jobs and JCL allocation for others!

13.3.5 RECON Data Set Placement
What follows are recommendations on placing the RECON data sets in order to reduce the possibility of having DASD contention problems:

- Place the three RECON data sets on different volumes with different control units, using different channels.
- Catalog the three RECON data sets in different user catalogs.
- Always use three RECON data sets, two active ones and a spare. In the INIT.RECON command, specify NONEW to avoid jobs starting when fewer than two RECON data sets are available.
- Put the ICF catalog, the BCS catalog, and the VVDS for each RECON data set on the same volume as the RECON data set. When RECONS are opened, two enqueues are done, one on the RECON followed by one on the catalog. Having them on the same device eliminates a source of deadlocks. When RECONS are accessed, one enqueue is done on each of the two active RECONS.
- The ultimate solution to avoiding a deadlock situation is to have the RECON as the only data set on a volume. This is suggested for production environments, but may be regarded as excessive for test and development systems.

Allocate the RECON data sets on DASD volumes that do not contain data sets that can be shared by other MVS images. If the DBRC data sets are allocated on volumes that can have a hardware RESERVE issued against them, the RESERVE does not allow other MVS images to access the DASD volume.

When the RECON data set is not on shared DASD, the MVS reserve process works like a simple ENQ and eliminates unnecessary I/O at the start of each call to the RECON for a single CPU environment. If the MVS systems generation identifies the DASD as shared, DBRC treats the volume as shared and will still process I/O to the device at the start of each call to the RECON, even in a single CPU environment.
**GRS Utilization:** When an IMS subsystem or DBRC utility accesses the RECON data sets, DBRC issues reserves against RECON data sets in DDname (RECON1,RECON2,RECON3) sequence. The reserves are issued using QNAME DSPURI01 and RNAME(DSN_FOR_RECONX). By default, IMS issues a hardware reserve and so locks out access to the entire volume from other requestors located on different MVS images. Other requestors on the same MVS image, will be allowed access to the volume. The reserves result in DBRC holding an EXCLUSIVE ENQ on the RECONs so only one instance of DBRC can use the RECONs.

To eliminate any potential shared DASD hardware reserve deadlock situations, global resource serialization (GRS) can be used to convert the reserve requests for the RECON data sets from hardware reserves to software enqueues (SYSTEM ENQs) and to propagate the SYSTEM ENQ request to all MVS images in a sharing complex.

The hardware reserve is cheap in terms of resources, but it locks out access to any other data set on the volume. A software ENQ is expensive, as this is a global ENQ, but it allows other data sets on the volume to be accessed. The cost varies by the number of MVS images in the sharing complex and by installation-specified tuning parameters, as described in GRS manuals.

The major name for RECON data sets is DSPURI01. It represents the generic entry that you have to add regarding your choice in one of the resource name lists (RNL).

The options are:

- If you are concerned about the performance of IMS, include DSPURI01 in the exclusion RNL. This generates a local system ENQ and a hardware reserve.
- If you are not so concerned about IMS performance, for example on a test system, and need to place other data sets on the same volume as the RECON, use the conversion RNL. This generates a global system ENQ, but no hardware reserve.

**13.3.5.1 Deadlock Situation**

It is possible to get into a shared DASD deadlock situation if a RECON data set and other VSAM data sets reside on the same volume and the data sets are cataloged in an ICF catalog that resides on a different volume. The deadlock situation can follow this scenario:

1. In System A, DBRC issues a RESERVE for a RECON data set. The RESERVE has a QNAME of DSPURI01, A hardware reserve is issued for the volume containing the RECON data set and the reserve is successful. System A now owns the RECON volume; that is, no other systems can access that volume until a hardware release is issued to it.

2. In System B, a normal VSAM OPEN is issued for a VSAM data set. VSAM OPEN processing issues a RESERVE for the catalog (BCS) that has the entry for the data set, and the reserve is successful. System B now owns the BCS volume. VSAM OPEN then issues a RESERVE for the VVDS on the RECON volume. A hardware reserve is issued to the volume containing the VVDS. Because this volume is owned by System A, System B waits for the volume. This wait lasts until the hardware release is issued to the volume from the system that issued the hardware reserve.
3. In System A, DBRC now issues a VSAM OPEN for the RECON data set. VSAM OPEN processing issues a RESERVE for the catalog (BCS). Because System B owns the BCS volume, (item number 2 above), System A waits until the hardware release is issued.

4. The result is that System A is waiting for a resource owned by System B, and System B is waiting for a resource owned by System A. Both tasks are now involved in a deadlock that can be resolved only by canceling one of the tasks.

The way to avoid this problem is to allocate the catalog (BCS) that contains the RECON data set to the same volume on which the RECON data set is allocated. Because the RECON data sets come in sets of three, this means that multiple catalogs (two or three) are required for the RECON data sets.

If all the RECON data sets are cataloged in the same ICF catalog, the loss of the catalog causes all of the RECONs to be lost. By using individual catalogs, only one RECON is lost when a catalog is lost — a definite advantage.

13.3.5.2 DASD Caching and DASD Fast Write
DASD caching and DASD Fast Write are two facilities that can be used to improve the I/O to the RECON data sets. It is recommended that the RECON data set reside on a cached DASD controller (3990-6 for example). If the DASD controller has no cache (and only in that case), then specify REPLICATE and IMBED in the VSAM DEFINE for RECON data sets.

13.3.5.3 RECON Reorganization
A way to reorganize a RECON data set is to use the CHANGE.RECON command. Do this after performing a DELETE.LOG INACTIVE.

Refer to the IMS/ESA V5 Utilities Reference: System (SC26-8036) for more information about this command.

13.4 RECON Access — Performance Considerations
DBRC functions can be very I/O intensive and are serialized by access to the RECONs. Bear this in mind when considering the following aspects of DBRC usage.

13.4.1 IMS Commands
The interaction between IMS and DBRC when processing a command is best illustrated by an example. Figure 96 on page 312 shows the effects of a /DBR DATABASE DB1 DB2 command.
### 13.4.2 DBRC Commands

As DBRC puts more and more information in the RECON data set, the DBRC LIST command is often used in conjunction with the DBRC GENJCL command to manage production IMS systems. An example is submitting an image copy just for the databases that have been modified rather than for all databases.

By using the GENJCL.USER command, it is easy to get information from the RECON and create the required jobs. The negative effect is the increased access to the RECON data set, which causes I/Os as well as RECON reserves.

It is always advisable to use specific operands on all list commands; the cost of a LIST ALL command can be very high.

### 13.4.3 Batch versus Online

Because DBRC is a single-threaded process, care must be taken to avoid causing IMS online processing to wait for IMS batch DBRC processing. This contention may have a performance impact on the IMS online workload, for example, causing an online system OLDS switch to wait.

### 13.4.4 DMB Pool Size

If the PRILOG record, shows that the ALLOC flag on each DBDS line is greater than one, several allocation/deallocation processes have occurred. This could be because of /DBR commands or because the DMB pool is too small. If the size is larger than what is expected from the number of /DBR commands, it needs to be investigated. The DMB pool must be large enough to contain database entries for all the IMS databases.

Following any DMB pool shortage, some databases must be closed to free space, which means deallocated in IMS as well as in DBRC. This results in a lot of DBRC processing, not only at cleanup time, but also at reopen time.
13.4.5 RECON I/O Exit (DSPCEXT0)

Using this exit routine enables you to keep track of changes to the RECON data set in the form of a journal. The journal is updated every time a record of the data set is updated, inserted, deleted, or read.

While this routine is running, the RECON data set is reserved so that no other jobs can access RECON records. Therefore, the best way to avoid performance degradation is to simplify the routine’s functions by limiting the I/O operations that it performs.

Refer to the IMS/ESA V5 Customization Guide (SC26-8020) for more information about this routine.

13.5 DBRC Buffer Allocation

To improve DBRC performance by reducing the number of RECON I/Os, DBRC buffer management uses a VSAM local shared resources pool. This allows DBRC to take advantage of lookaside and least recently used techniques.

The DBRC buffer pool is located in the DBRC address space for IMS DB/TM and DBCTL subsystems, and in the batch address space for an IMS batch subsystem. The buffer pool is above the 16 MB line. The LSR pool ID used for the RECON buffers is 15. If this pool ID is already used by another product, DBRC tries to use another one, going from 14, 13 ... down to 1.

The number of buffers is specified in the DSPBUFFS CSECT, which is link-edited into the DSPCINT0 module. Figure 97 shows the provided DSPBUFFS CSECT that a user may modify to increase the number of LSR buffers used for RECON processing.

```
DSPBUFFS CSECT , DECLARE NBR OF INDEX & DATA BUFFERS
DSPBUFFS AMODE 31
DSPBUFFS RMODE ANY
   DC CL8'DSPBUFFS' REQUIRED EYECATCHER FOR DUMPS
   *
   LSRONLIN DC AL2(6,12) XA ENVIRON - IMS ONLINE DBRC
   LSRCICS DC AL2(6,12) XA ENVIRON - CICS USE OF DBRC
   LSRBATCH DC AL2(6,12) XA ENVIRON - OFFLINE/BATCH DBRC
   *
   NSRONLIN DC AL2(2,2) NONXA ENVIRON - IMS ONLINE DBRC
   NSRCICS DC AL2(2,2) NONXA ENVIRON - CICS USE OF DBRC
   NSRBATCH DC AL2(2,2) NONXA ENVIRON - OFFLINE/BATCH DBRC
   END DSPBUFFS
```

Figure 97. DSPBUFFS CSECT

Each of the six data declarations defines a pair of numbers. The first is the number of index buffers, the second is the number of data buffers.

The NSR buffer space is required only during the initial physical open process. The open for NSR mode requests two index and two data buffers. This results in 20 KB of buffers per data set being allocated during the NSR open process. These buffers are freed when the RECONs are closed during the physical open process.
Do not specify the BUFFERSPACE, BUFNI, or BUFND parameters when allocating the VSAM clusters.

The *IMS/ESA V5 Customization Guide (SC26-8020)* describes the buffer allocation process and gives an example of a DSPBUFFS change.

The default value is not sufficient; at a minimum, double it for the index component and make it four times as big for the data component. The recommended buffer numbers for the DB/TM system are defined with:

```
LSRONLIN DC AL2(12,48)
```
Chapter 14. IMS Database Control

The IMS Database Control (DBCTL) facility provides access to an IMS Database Manager Subsystem for multiple CICS regions and BMP jobs, allowing them concurrent access to IMS full-function and DEDB databases.

The CICS/ESA V4.1 CICS—IMS Database Control Guide (SC33-1184) contains all relevant information from a CICS perspective. The standard IMS documentation contains all the details on implementing and managing a DBCTL environment.

This chapter focuses on the performance benefits and issues of DBCTL. A DBCTL system is an IMS DB/TM system with no IMS terminals. The IMS information contained in this book, excluding the IMS/TM specifics, is applicable to DBCTL.

Local DL/I support is not supported in IMS 5.1. Any mention here of CICS - Local DL/I is simply for comparative or migration-related reasons only.21

Analyzing performance in the DBCTL environment has three facets:
• CICS tuning — covered in the CICS/ESA V4.1 Performance Guide (SC31-1183)
• DBCTL tuning — contained in this book
• Interface tuning — contained in this book

The link between CICS, DRA, and DBCTL is called the recovery token. It is available in traces, messages and dumps for use in problem resolution.

14.1 CICS to DBCTL Interface and Interface Components

This section defines the components of the interface between the CICS address space and the IMS DBCTL subsystem address space.

Figure 98 on page 316 shows the relationship between CICS (and its components) and IMS DBCTL (and its components). The CICS interface component to IMS, called DRA, resides inside the CICS online region. It, however, is developed and shipped with IMS.

The coordinator control (CCTL) subsystem is the transaction management component that communicates through the database resource adapter (DRA) to DBCTL. In a CICS-DBCTL environment, CICS is the CCTL.

21 Migrating to DBCTL from the local DL/I environment must take place on IMS Version 3.1 or 4.1, prior to converting to IMS 5.1.
Within CICS, the interface has three major components:

- The CICS-DL/I Router (DFHDLI), is an interface between application programs and the CICS DL/I Call processor (DFHDLIDP)
- The CICS Database Adapter Transformer (DFHDBAT) communicates with the DRA to enable connection and disconnection to DBCTL, allowing DL/I requests to be processed
- The Database Resource Adapter (DRA) establishes and manages the connections between CICS transactions and DBCTL processing

14.2 The DBCTL High Performance Environment

This section discusses the implicit performance benefits of a CICS-DBCTL system compared with the previous CICS-Local DL/I environment.

14.2.1 Unconstrained Virtual Storage

Many CICS-Local DL/I customers experienced a virtual storage constraint which, in many cases, imposed performance constraints. DBCTL removes almost all IMS-related virtual storage from the CICS address space. More specifically, this includes:

- DL/I code and control blocks
- DBRC code
- OSAM and VSAM buffer pools and related control blocks
- PSB, DMB, and ENQ/DEQ buffer pools
A major advantage of converting to DBCTL for many CICS users concerns DBRC. Previously, it was often impossible to implement DBRC because of below-the-line storage constraints. With the VSCR that DBCTL brings, you are now free to implement DBRC to protect and manage the recovery of your databases.

**Application Use of Storage Above 16 MB Line**

1. In EXEC DLI programs, both the program and the data can reside above the 16 MB line. This requires the specification of AMODE=31 when the program is link-edited.

2. In CALL DLI programs, CICS online programs accessing a remote database and programs controlled by DBCTL can use AMODE(31) when a transaction is defined with TASKDATALOC(ANY).

**14.2.2 Parallel DL/I Environment**

DBCTL provides a separate task control block (TCB) for each application thread from CICS. This approach is very much like the IMS TM parallel DL/I environment with LSO=Y. An IMS TM system is limited to 999 dependent regions; a DBCTL system is likewise limited to 999 threads or BMP regions.

If, for example, a CICS system has 20 threads with DBCTL, then 20 CICS transactions can each be executing a DL/I call concurrently. This is a major improvement over Local DL/I, where only one DL/I call, running under a single CICS TCB, can execute at a time.

**14.2.3 Access to Data Entry Databases**

DEDBs are designed for high performance as well as very large database support and high availability. Local DL/I users are unable to take advantage of DEDBs.

DBCTL gives CICS transactions the opportunity to use DEDBs, and so gain their performance benefits. The following restrictions do, however, apply to DEDBs:

- No stand-alone batch support (BMPs must be used for batch work)
- No secondary indexes
- No logical relationships

From a performance perspective, DEDBs provide:

- A very quick and low-cost facility for data entry (the sequential dependent segment or SDEP).
- An efficient way to enter and retrieve data in roots and direct dependent segments.
- An efficient way to gain access to segments in the middle of twin chains (subset pointers).
- A reduced path length by using MVS/DFP Media Manager.
- Reduced logging and no forced log writes.

---

22 CICS online programs that access a Local DL/I database (prior to IMS V5) require that the DL/I call parameter list, and all storage areas that are referred to in the parameter list, reside below the line. In CICS/ESA, you must specify TASKDATALOC(BELOW) for such transactions. In addition, you may choose to specify AMODE(24) when you link-edit the program.
Even more parallelism by using asynchronous parallel write I/Os.

A sequential buffering facility to provide high-speed sequential processing (HSSP). This also allows an image copy to be taken in the same pass through the database.

A high-speed online reorganization utility.

14.2.4 Using BMPs for Parallel Batch Processing

Stand-alone batch programs are unaffected by what facility is used to provide online data access. However, more and more customers need to provide access to batch and online systems at the same time.

CICS provides a shared database support facility called the *interregion communication* (IRC). It was intended to be the way to share a database between CICS and batch jobs with data integrity. With DBCTL, we recommend that the shared batch jobs be converted to BMPs, which provide the following benefits:

- Transaction response times and throughput are no longer seriously impacted by batch processing.
- The batch jobs run more quickly.
- Programs can be link-edited with AMODE(31).
- Programs can exploit IMS checkpoint and restart facilities such as symbolic checkpoint (CHKP) and extended restart (XRST) calls.
- Access to sequential files can be replaced with DL/I access to GSAM files. The benefit is that IMS takes complete responsibility for repositioning GSAM files at BMP restart time.
- DBCTL uses dynamic backout of failed BMPs, quickly releasing locks and expediting program restart.
- A single log is produced by DBCTL.
- Only one set of buffers need to be tuned (the DBCTL buffers).

The batch to BMP conversion is not mandatory in a DBCTL environment, but it must be considered as a solution to certain operational problems. It is the preferred solution for running parallel batch. In a Parallel Sysplex environment, multiple stand-alone batch jobs can share databases. However, they also need to exploit CHKP/XRST and GSAM and are limited to a maximum of 32 concurrent jobs across all MVS systems in the sysplex. Each batch job is considered to be an individual subsystem, whereas BMPs under DBCTL are considered to be parts of a single subsystem.

Note, however, that a BMP does have a longer path length than a stand-alone batch job. This must be weighed against all the advantages that are gained from using BMPs.

14.2.5 Potential for Reduced CPU Cost

Where a single CICS system is used to process a whole transaction (in other words, one system acts both as application-owning and data-owning region) an increase in transaction path length is experienced in migrating to DBCTL. This is caused by the fact that each DL/I call to DBCTL involves two MVS task switches. Clearly, the cost is proportional to the number of DL/I calls issued.
Prior to DBCTL, however, a common environment was to have a data-owning CICS region using Local DL/I, and one or more application-owning regions. The CICS multiregion option is exploited to ship DL/I calls from one region to the next.

When converting to a DBCTL environment, the data-owning region is replaced by the DBCTL subsystem. In this case, a DL/I call has a significantly shorter path length in the DBCTL environment than in the CICS multiregion option environment. Again, the overall CPU reduction is proportional to the number of DL/I calls issued.

### 14.3 CICS—DBCTL Performance Monitoring

Performance monitoring in a CICS-DBCTL environment involves collecting information from several sources:

- SMF statistics from CICS sources
  - CICS Monitoring Facility Performance report for each task
- RMF statistics on IMS processing
- IMS log record information
- IMS Monitor information
  - Using /TRACE SET ON MONITOR command
  - Collected in the IMSMON DD card data set
- IMS trace information
  - Using /TRACE SET ON TABLE command
  - Collected on the IMS log or external trace data set

### 14.3.1 CICS/DBCTL Events

Some of the key events in the life of an IMS transaction that are discussed elsewhere in this book do not occur in a DBCTL environment (they are CICS functions rather than IMS functions). Some events occur in both environments but have a different cause with CICS-DBCTL. The basic events within DBCTL (DRA requests) in the life of a CICS transaction are:

1. A SCHED request for a PSB to be scheduled
2. DL/I requests to make database calls
3. A sync point request (COMMTERM) to drive sync point processing and release the PSB

### 14.3.2 IMS /DISPLAY Commands

Prior to IMS 5.1 and CICS 4.1, the only way to submit IMS commands in a DBCTL environment was to use the MVS console.

CICS 4.1 provides a transaction, CDBM, which enables DBCTL operator commands to be entered from a CICS session. CDBM uses the Type 2 AOI facility in IMS 5.1. For information on this facility, refer to the standard IMS documentation and the ITSO IMS redbooks list in Appendix E, “Related Publications” on page 391.
To display the current status of various DBCTL resources, the following commands are available (through CDBM or from the MVS console):

```
/DISPLAY ACTIVE
/DISPLAY CCTL
/DISPLAY POOL
/DISPLAY STATUS
/DISPLAY TRACE
```

**Note**

It is recommended that you use the DBCTL execution parameter PREMSG=N. This suppresses the DFS000I message that precedes many DBCTL system messages displayed on the MVS console.

### 14.3.2.1 IMS Log Statistics

IMS writes certain performance information to the log records. IMS appends the following information to the X'08' log record during PSB allocation:

- Total elapsed wait time because of intent conflict
- Total elapsed wait time because of pool space not being available
- Total elapsed time for a schedule request

Similarly, IMS appends the following information to the X'07' log record at PSB termination:

- Total number of databases involved in I/O
- Total number of DL/I database requests
- Total elapsed wait time as a result of locking
- Total number of gets
- Total number of inserts
- Total number of replaces
- Total number of deletes

### 14.3.2.2 Trace Options

Trace options are shown in Figure 99 on page 321.

**CICS Trace:** The CICS-supplied transaction CETR controls CICS tracing activity. It can trace the DL/I activity, including DBCTL, to the point where it leaves the CICS database adapter transformer (DFHDBAT). For more information on CETR, refer to *CICS/ESA V4.1 CICS-Supplied Transactions (SC33-1168).*

**IMS Trace:** An IMS TRACE command is available in DBCTL for tracing internal IMS events. This command is also used to start, stop, and define the activity monitored by the IMS Monitor. To start tracing, use either the DFSVSMxx member in PROCLIB or the IMS /TRACE command.
* For Online tracing to in-core trace tables or to an external file
  /TRACE SET ON TABLE XXXX  OPTION LOG|NOLOG
  with XXXX = ALL, DISP, DL/I, DLOG, FAST, LATC,
  LOCK, LRTT, RETR, SCHD, STRG, SUBS

* For Program Isolation Tracing
  /TRACE SET ON PI

* For IMS Monitor
  /TRACE SET ON MONITOR ALL|APDB|SCHD

Figure 99. IMS DBCTL Trace Options

For full-function DL/I databases, you can use the PI trace to collect information about all PI enqueue/dequeue activity. A report of all PI contentions is produced using the PI trace report utility (DFSPIRP0). (The listing of all the trace records can be produced with the log formatting and print utility, DFSERA10, using the DFSERA40 exit.) See the IMS/ESA Administration Guide: System for further guidance on using the PI trace.

**IMS Monitor:** In a DBCTL environment, the IMS monitor reports for message queues, MFS, communications, and other message-related activities are either not produced or are empty. The applicable reports are exactly as for an IMS transaction manager system, and so require some interpretation:

<table>
<thead>
<tr>
<th>IMS Term</th>
<th>CICS-DBCTL Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region or PST</td>
<td>CICS thread or BMP region</td>
</tr>
<tr>
<td>Program</td>
<td>PSB</td>
</tr>
<tr>
<td>Transaction</td>
<td>PSB scheduling</td>
</tr>
</tbody>
</table>

**Note**

When DBCTL is servicing multiple CICS systems, the region, as reported by IMS Monitor, can represent different threads from different CICS systems over the monitoring period.

The most relevant IMS Monitor reports for a DBCTL system are likely to be the program-related reports:

- Call summary
- Program I/O
- Database buffer pool (such as OSAM)
- VSAM buffer pool
- Program summary

The IMS monitor reports relating to threads may also be of interest:

- Region summary
- Region IWAIT
  - Scheduling IWAITs (including ACBLIB I/O)
14.3.2.3 CICS-IMS Statistics

DBCTL supplies CICS with statistical information at disconnect time. These statistics, known as unsolicited statistics, are specifically passed to CICS (when possible) at:

- An orderly or immediate disconnection of DBCTL
- An orderly termination of CICS

CICS-DBCTL statistics are not collected if there is an immediate shutdown or abend of CICS.

The actual collection process is as follows:

1. The DRA returns statistics for the terminating CICS-DBCTL session to DFHDBAT.
2. DFHDBAT invokes the CICS statistics exit for DBCTL statistics (DFHDBSTX).
3. DFHDBSTX invokes the CICS statistics domain.
4. The CICS statistics domain collects the statistics.

CICS-DBCTL session statistics are contained in the DFHDBUDS DSECT, which you can generate from the copybook DFHDBUDS. The actual values involved are:

- DBCTL identifier for the CICS-DBCTL session (STATDBID)
- DBCTL recoverable service element (RSE) name (STARSREN)
- Time at which CICS connected to DBCTL (STACTIME)
- Time at which CICS disconnected from DBCTL (STADTIME)
- Minimum number of threads specified in the DRA startup table (STAMITHD)
- Maximum number of threads specified in the DRA startup table (STAMATHD)
- Number of times that the CICS-DBCTL session collapsed threads down to the minimum thread value specified in the DRA startup table (STAMITHD)
- Number of times that the CICS-DBCTL session reached the maximum thread value specified in the DRA startup table (STAMATHD)
- Elapsed time, expressed in hours, minutes, and seconds, for which the CICS-DBCTL session ran at the maximum thread value (STAELMAX)
- Largest number (also known as the high-water mark) of threads used during the CICS-DBCTL session (STHIWAT)
- Total number of times the CICS-DBCTL session successfully scheduled a PSB (STAPBSU).

To extract and print a report from these statistics, the CICS-supplied statistics utility program (DFHSTUP) is run, specifying the specific APPLID of the relevant CICS system. The output includes CICS-DBCTL session statistics, provided DBCTL was connected to CICS when the statistics were collected. For guidance on other parameters needed to run DFHSTUP, and a sample job stream, see the CICS/ESA
V4.1 Operations and Utilities Guide (SC33-1167). Figure 100 on page 323 shows a sample report produced by running DFHSTUP.

**DBCTL SESSION TERMINATION STATISTICS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCTL SESSION NUMBER</td>
<td>1</td>
</tr>
<tr>
<td>DBCTL IDENTIFIER</td>
<td>SYS1</td>
</tr>
<tr>
<td>DBCTL RSE NAME</td>
<td>DBCTLSY1</td>
</tr>
<tr>
<td>TIME CICS CONNECTED TO DBCTL</td>
<td>13:33:27.2619</td>
</tr>
<tr>
<td>TIME CICS DISCONNECTED TO DBCTL</td>
<td>15:52:46.3777</td>
</tr>
<tr>
<td>MINIMUM NUMBER OF THREADS</td>
<td>1</td>
</tr>
<tr>
<td>MAXIMUM NUMBER OF THREADS</td>
<td>10</td>
</tr>
<tr>
<td>TIMES MINIMUM THREADS HIT</td>
<td>2</td>
</tr>
<tr>
<td>TIMES MAXIMUM THREADS HIT</td>
<td>5</td>
</tr>
<tr>
<td>ELAPSED TIME AT MAXIMUM THREADS</td>
<td>00:02:46:5090</td>
</tr>
<tr>
<td>PEAK NUMBER OF THREADS</td>
<td>10</td>
</tr>
<tr>
<td>SUCCESSFUL PSB SCHEDULES</td>
<td>146</td>
</tr>
</tbody>
</table>

Figure 100. DBCTL Session Termination Statistics

**Note**

Elapsed time at maximum number of threads shows a significant amount of time when no more threads were available. This could well mean that transactions waiting to be processed were held up for lack of access to IMS resources. It appears that the maximum number of threads needs to be increased.

Statistics written to SMF data sets are collected only if explicitly specified. This is done by issuing the CEMT SET STATISTICS command from the master terminal or by running a program containing the command EXEC CICS SET STATISTICS (see CICS/ESA V4.1 CICS-Supplied Transactions (SC33-1168) and CICS/ESA V4.1 System Definition Guide (SC33-1164)).

14.3.2.4 Global User Exits XRMIIN and XRMIOUT

In CICS 4.1, the global user exits XRMIIN and XRMIOUT enable the monitoring of activity across the resource manager interface. They aid in monitoring the DL/I activity. Refer to the CICS/ESA V4.1 Customization Guide (SC33-1165) for more information.

14.4 Performance Parameters in CICS and IMS

System design considerations for CICS with DBCTL are similar to those for Local DL/I. However, some differences do exist. DBCTL is structured to have one TCB per thread. This allows for concurrent processing. However, the minimum and maximum numbers of threads that are consistent with the system’s requirements must be specified.

The storage specified in the CICS system-initialization parameters CDSASZE/CDSALIM and EDSALIM is used for different resources in a CICS-DBCTL environment. In all environments, CDSASZE/CDSALIM is used to specify the size of the dynamic storage area (DSA) below 16 MB, and EDSALIM is used to specify the size of the extended dynamic storage area (EDSA) above 16.
MB. Local and remote DL/I use DSA for PSB and DMB pools, but with DBCTL, these blocks are stored outside CICS. Instead, you need to allow for the storage DBCTL needs in CICS for database resource adapter (DRA) code when specifying EDSALIM. See the CICS/ESA V4.1 System Definition Guide (SC33-1164) and the CICS/ESA V4.1 Performance Guide (SC33-1183) for guidance on specifying EDSASZE, and the IMS/ESA V5 Administration Guide: System (SC26-8013) for guidance on DBCTL storage estimates.

From an IMS point of view, tuning DBCTL is much like tuning an IMS system. Additional consideration, though, must be given to DRA threads and DEDB buffer pool allocation.

### 14.4.1 DL/I Thread Allocation

The CICS to DBCTL communication is handled by threads. Each thread represents a communication path between the CICS transaction and DBCTL. In the DBCTL address space, the thread identifies the CICS transaction's existence and traces its progress and its access to IMS resources.

### 14.4.2 DRA MINTHRD and MAXTHRD Parameters

The DRA startup parameter table provides the parameters needed to define the interface to the DBCTL subsystem. It is created by assembling the DFSPRP macro and link-editing it into the IMS.RESLIB library (or another APF-authorized library) with the member name DFSPZPxx.

The DRA startup parameters MINTHRD and MAXTHRD specify the minimum and maximum number of threads that can process DBCTL DL/I or DEDB requests concurrently.

Specify a MINTHRD large enough for your system's typical needs. However, if this value is higher than your system's actual minimum requirement, it can sometimes tie up threads unnecessarily, preventing DBCTL from allocating them to other CICS systems or BMPs. On the other hand, if MINTHRD is specified too low, performance is also affected. As the level of thread activity falls, the DRA releases threads down to the minimum value. These threads then have to be reestablished when the CICS activity increases again.

In general, we recommend that the average number of threads in the peak hour be specified for MINTHRD. We also recommend that the number specified for MAXTHRD reflect the peak number of DBCTL threads required.

To help determine the optimum values for MINTHRD and MAXTHRD, monitor the following:

- Thread usage
- IMS task throughput (to see if tasks are being delayed)
- IMS I/O rates

Details of thread usage, including maximum and minimum usage, can be seen from the CICS Session Termination statistics.
14.4.3 IMS MAXREGN Parameter

The IMS system generation parameter MAXREGN specifies the minimum number of control blocks (dependent region PSTs) allocated permanently to IMS dependent regions or CICS threads. These blocks are allocated at IMS startup, but additional PSTs are created dynamically if necessary (up to the maximum of 999).

Specify a MAXREGN value for the DBCTL system that is at least the total of the MINTHRDs for all the CICS systems plus the number of BMP regions required.

In Figure 101, the following threads are in use: one from BMPA, one from BMPB, five from CICSA, and three from CICSB, making a total of ten threads. A MAXREGN of ten has therefore been specified for DBCTLA to create the necessary control blocks part of IMS initialization.

![Diagram of CICS/ESA-DBCTL Threads Requirements](image)

Figure 101. CICS/ESA-DBCTL Threads Requirements

14.4.4 CICS MXT SIT Parameter

Take DBCTL thread activity into account when specifying the MXT parameter in the CICS system initialization table. MXT specifies the maximum number of tasks that CICS allows to exist at any time. If DB2 is also used, make an allowance for the threads it needs as well. Define MXT large enough to support the number you specify in MAXTHRD plus the number needed for standard CICS tasks.

14.4.5 Storage Requirements

Each CICS-DBCTL thread uses 9 KB of LSQA below the 16 MB line and 3 KB of SP230.23

---

23 Threads for Local DLI were defined by the DLTHRED= parameter in the system initialization table and used 9 KB of OSCOR and 3 KB of DSA
Beware

CICS permanently allocates 5 KB of storage below 16 MB for each potential task, plus a corresponding number of 8 KB allocations above 16 MB. This is in addition to the amount of DSA and EDSA storage specified in the system initialization table. Thus, the higher the MXT, the less operating system storage is available below 16 MB for other purposes.

Also be aware that CICS obtains storage for the DSA before the storage for each task (using the MXT parameter). If a high DSASZE is specified, there may not be enough storage left in the private area below the line to satisfy the number of tasks in MXT. A balance between the MXT and DSAASZE needs to be found, leaving enough storage to satisfy other requests for operating system storage.

Refer to the *CICS/ESA V4.1 System Definition Guide (SC33-1161)* for guidance on specifying the DSASZE, EDSASZE, and MXT parameters. Also see the *CICS/ESA V4.1 Performance Guide (SC33-1183)* for general guidance on MXT and DSA, including its effects on a CICS system and how to monitor it.

14.4.6 IMS Database Buffer Pools

14.4.6.1 Full-Function Database Pools

For full-function databases, the buffer pool is located in the DLISAS address space. There is no special specification for CICS transactions because the OSAM and VSAM pools are shared among all the DLI requesters.

14.4.6.2 Fast-Path DEDB Buffer Pool

Within the DBCTL system, the standard control region execution parameters are used to define the size of the Fast Path buffer pool (DBBF) and the number of buffers to be immediately page-fixed (DBFX).

When each region in an IMS DB/TM system starts up, the NBA execution parameter is used to page fix additional buffers and make them usable. A similar mechanism is needed to page fix buffers for each CICS thread.

When a CICS thread connects to IMS, its DEDB buffer requirements are taken from the DRA startup table. These parameters are:

- **CNBA** — total number of buffers needed for the CICS system for normal buffer allocations.
  
  A CICS system cannot connect to DBCTL if its CNBA value is more than the number of buffers remaining in DBBF.

- **FPBUF** — number of buffers to be available as normal buffer allocation for each thread (equivalent of NBA).

  FPBUF is a subset of CNBA. An application attempting to schedule a PSB that contains one or more PCBs for DEDBs receives a schedule failure if the FPBUF value is more than that currently available from CNBA.

- **FPBOF** — number of overflow buffers a thread can use if it exceeds FPBUF (equivalent of OBA).

  When a CICS system has successfully connected to DBCTL, and the application has successfully scheduled a PSB containing DEDB PCBs, the
FPBOF parameter becomes relevant. These buffers are not taken from CNBA. Instead, they are serially shared by all CICS applications or other dependent regions that are currently exceeding their FPBUF or NBA limits. In other words, only one thread or region can use its overflow buffers (OBA or FPBOF) at a time.

In summary, the Fast Path buffer pool contains FPBUF buffers in virtual storage. However, only page-fixed buffers can be used. The page-fixed buffers are:

- DBFX (system defined)
- CNBAs for all CICS systems
- NBAs for all BMPs
- Largest current FPBOF/OBA
- One for each open area with SDEP defined in DBD

These page fixed buffers are available to any thread or BMP or for SDEP collection and are allocated as required. The maximum number that can be allocated to a thread for normal use is FPBUF, and to a BMP is NBA. In exceptional cases, a thread can use up to FPBOF more buffers (OBA for a BMP), but their use is serialized by the OBA latch.

The key difference between DBCTL and DB/TM buffer specifications is that each IMS dependent region can have a different NBA, whereas each thread within a CICS system has the same FPBUF. This means that FPBUF must be specified for the PSB that uses the most buffers.

If every PSB references DEDBs, then CNBA must be set as FPBUF × MAXTHRDs. If it is known that a maximum of n concurrent PSBs are to access DEDBs, then CNBA can be set as n × FPBUF.

## 14.5 MVS Considerations

Two main MVS aspects need to be considered: dispatching priority and CPU time.

### 14.5.1 Dispatching Priority

We recommend that a higher dispatching priority be assigned to the CICS address space than to the DBCTL address space, as CICS is the front-end of the CICS-DBCTL system. Assign to the DL/I SAS address space job dispatching priorities of one less than DBCTL, and assign the DBRC address space a dispatching priority that is one higher than CICS.

### 14.5.2 CPU Time

The CPU time for DBCTL is not accumulated in DFHTASK, because in this case the DL/I activity is not executing under a CICS TCB.

The DL/I CPU time is logged, as for a DB/TM system, in the X'07' log record at PSB termination. This time does not quite account for 100% of IMS CPU used. Control region activities such as physical logging and PSB scheduling are not captured in this way.
Chapter 15. Performance of IMS with DB2

Several DB2 attachment facilities are provided for different environments:

- The IMS external subsystem attach facility (ESAF)
- The IMS batch attach facility
- The CICS attach facility
- The TSO attach facility
- The call attach facility for any MVS batch

The IMS ESAF interface provides access to external subsystems from any supported region type (MPP, BMP, or IFP) in any IMS system environment (DB/TM, DCCTL, or DBCTL).

Theoretically, any MVS subsystem can implement support for this interface. The two MVS subsystems that currently support the IMSESASF are:

- DB2 for MVS/ESA (referred to as DB2)
- MQSeries for MVS/ESA (referred to as MQM)

This chapter describes the IMS ESAF and various aspects of performance relating to its use in accessing DB2.

15.1 IMS External Subsystem Attach Facility

The IMS ESAF enables users to access DB2 and MQM from IMS DB/TM, DCCTL, and DBCTL. Specifically for DB2, it provides support for:

- Connection between the IMS subsystem and one or more DB2 subsystems
- Communication between an IMS control region and DB2 for a command thread (for issuing DB2 commands in IMS)
- Communication between IMS dependent regions and DB2 for transaction threads (for enabling and processing SQL calls)
- The SQL application programming interface
- Coordinated sync point processing

IMS application programs using ESAF have concurrent access to the following resource managers:

- IMS message queue manager (online applications only)
- IMS full-function database manager
- IMS Fast-Path database manager (online applications only)
- DB2 database manager
- MQSeries queues (local and remote queues)
IMS sync point processing uses a two-phase commit. IMS assumes the role of sync point coordinator and communicates with all the relevant resource managers, including DB2. In case of a program or system failure, IMS coordinates the backout or recovery of both DB2 and IMS data (messages and databases).

IMS supports read-only threads, which improve the DB2 read-only commit processing by eliminating the unnecessary Phase 2 call.

The remainder of this chapter considers only the DB2 connection; however, the same concepts apply equally to MQM.

Figure 102 shows the relationship between DB2 and IMS.

IFP and MPP regions identify themselves to DB2 during the initialization phase of their startup. BMP regions wait until the first SQL call to identify themselves to DB2.

15.1.1 Subsystem Member

The IMS subsystem member execution parameter defines which subsystem member (SSM) in PROCLIB is read by IMS when it starts up. The SSM defines which DB2 subsystems can communicate with this IMS. Each dependent region also has an SSM execution parameter, indicating which PROCLIB member defines the DB2 subsystems with which this region can communicate. This can be the same set or a subset of those defined for the control region.

24 For a CICS DBCTL transaction, CICS is the sync point coordinator. For a BMP in a DBCTL environment, IMS is the coordinator.
The member used by the IMS control region is the default for the dependent regions, which do not have any SSM specification.

If some regions have no need to access DB2, use an SSM that exists but is empty. Each entry has the following positional parameters:

- SSN — DB2 subsystem name
- LIT — language interface token
- ESMT — external subsystem module table (DSNMIN10 is the required value)
- RTT — user-generated resource translation table (optional)
- REO — region error option
- CRC — command recognition character

Refer to the DATABASE2 V4 Administration Guide (SC26-3265) for further details.

Figure 103 shows the relationship between the SSM and dependent region.

15.2 Tuning the External Subsystem Attach Facility

The guidelines for using ESAF are:

- Minimize the number of thread allocations and terminations.
- Minimize the duration of locks.
• Avoid defining more external subsystems than needed in the SSM. Each
definition adds storage requirements. Use an empty SSM for regions that do
not require access to DB2, otherwise the control region SSM is used.

• Consider isolating DL/I applications in IMS dependent regions separate from
SQL applications. This facilitates:
  – Performance monitoring and problem isolation
  – Selection of scheduling algorithms and class assignments.

• Analyze applications to choose between static SQL and dynamic SQL.

• Choose the best security level according to installation requirements.

• Choose accurate dispatching priority for subsystems.

15.2.1 Thread Management

Thread creation and termination can use significant amounts of CPU in an
IMS-DB2 transaction. To reduce the CPU cost of thread creation at program
schedule time (and at signon to DB2), IMS 5.1 includes a design change that uses
a program call instead of an SVC call and makes use of cell pool (CPOOL)
requests to allocate dynamic storage. The IMS monitor, in the region IWAIT report
(see Figure 107 on page 341), shows the time of the thread creation and
termination. It reports them as IWAITs, but they are not I/O IWAITs, although the
SQL normal call values do, of course, include I/Os. The simpler the transaction,
the proportionally higher the cost of thread creation and termination. Consequently,
CPU time can be saved if the number of thread allocations (and hence eventual
terminations) are kept to a minimum. Suggestions for achieving this are:

• Use a specific number of regions connected to the DB2 subsystem.

• Keep threads running as long as possible by exploiting:
  – Wait-for-input transactions
  – Pseudo-wait-for-input
  – Quick reschedule

Fast Path (IFP) regions, being wait-for-input create the thread only once.

Design for thread reuse as much as possible to spread the cost of thread creation
and termination over multiple transactions. Collapsing multiple MPPs into a single
MPP, if practical, can help.

---

**Note**

The thread create and terminate processes are executed under the caller's
TCB, and the CPU time is charged to the IMS dependent region.

15.2.1.1 Wait-for-Input

The IMS WFI function is the ideal solution for minimizing thread allocations, as it
happens just once when the program is scheduled. However, a region occupancy
of more than about 70% is required to fully justify its use.
15.2.1.2 Pseudo-Wait-for-Input

Pseudo-WFI (PWFI) applies to all message processing programs (MPPs)25, and IMS DB2 applications can benefit not only from reduced program load but from reduced DB2 thread creation as well.

PWFI is discussed in 10.2.4, “Pseudo-WFI Transactions” on page 210.

15.2.1.3 Quick Reschedule

Although this is not an option from IMS’s perspective, there are two requirements for its use:

- The application program must issue a GU call against the I/O PCB after each message is processed, to drive sync point processing. It should terminate on a QC status code only.
- PROCLIM > 0 must be specified on the TRANSACT macro.

If the program is held, waiting on the IOPCB GU, and another message arrives for that transaction, it can be immediately passed to the program, so that the thread to DB2 from the previous scheduling is still available.

Quick reschedule is described in 10.2.5, “Process Limit Count and IMS Quick Rescheduling” on page 211.

15.2.2 DB2 Lock Management

Locking considerations are important for the system’s throughput and performance. Attention must be paid to when a lock is acquired, which level of lock is used, and when the lock is released.

Pages, tables, or table spaces may be locked during an application process in a manner similar to the enqueue/dequeue (ENQ/DEQ) for DL/I. The page locks result from SQL calls and can be compared to the IMS program isolation (PI) locks.

15.2.2.1 LOCKSIZE Parameter

When creating a table space, you choose the granularity of locking by specifying the LOCKSIZE:

- LOCKSIZE(PAGE)

  Page locking is the best strategy for concurrency in an online environment. In general, it is achieved by specifying LOCKSIZE(ANY). However, for WFI transactions, you should specify LOCKSIZE(PAGE) explicitly.

- LOCKSIZE(ANY) is the default and is recommended. It implies a page-level locking strategy. However, if too many page locks are taken for a table, lock escalation occurs so that a single table-space lock is acquired instead.

  For online processing, lock escalation is undesirable. If locking problems occur with LOCKSIZE(ANY) because of lock escalation, review the application design, and take sync points much more frequently.

  The exception to the recommendation is for WFI transactions, which should use table spaces defined with LOCKSIZE(PAGE). If ANY is specified, and lock

---

25 When PWFI was first introduced, IMS DB2 applications were specifically excluded. IMS 5.1 enables PWFI to be exploited by all MPPs.
escalation takes place, then locking remains at the table space level until program termination (not transaction termination); no deescalation facility exists.

- **LOCKSIZE(TABLESPACE)** or **LOCKSIZE(TABLE)** are not recommended in an online environment.

- **LOCKSIZE(ROW)**
  Row-level locking was introduced in DB2 4.1. For normal transaction processing, no benefit is generally provided by locking rows rather than pages. Therefore, its use is not recommended unless it is the only way to suppress locking conflicts.

### 15.2.2.2 INDEX Definition

The Type 2 indexes, which became available in DB2 4.1, are highly recommended. With the older type of index, most lock contention typically occurs in the index tree. With Type 2 indexes, no locks are taken in the index tree.

### 15.2.2.3 BIND ACQUIRE Parameter

This parameter determines when table-space intent locks are acquired.

- With **ACQUIRE(ALLOCATE)**, all locks are taken when the PLAN is allocated. This corresponds to when the thread is created, which in turn occurs at the first SQL call after IMS program scheduling.

- With **ACQUIRE(USE)**, each table-space intent lock is acquired individually, as each table space is first referenced by the application.

If most of the tables included in a PLAN are actually referenced at each execution, then **ACQUIRE(ALLOCATE)** is best. A locking problem with any table space is recognized before any work is done. With **ACQUIRE(ALLOCATE)**, the first transaction pays the full cost. **ACQUIRE(ALLOCATE)** is useful for IMS WFI transactions.

If some tables are not referenced in every execution, then **ALLOCATE(USE)** is the most efficient technique and is the recommended default for non-WFI transaction programs.

#### Note

If you are using DB2 packages, you cannot choose. **ALLOCATE(USE)** is forced.

### 15.2.2.4 BIND RELEASE Parameter

This parameter determines when table space locks are released. (Page locks or row locks are, of course, released during message processing or at sync point time, exactly as for IMS full function PI locks.) The **RELEASE** parameter can have a significant effect on performance:

- With **RELEASE(DEALLOCATE)**, the table-space locks are released when the PLAN is deallocated (in other words, when the IMS program terminates)

- With **RELEASE(COMMIT)**, the table-space locks are released at each sync point.

**RELEASE(COMMIT)** can have a severe performance impact as plan resources (table space locks, cursor sections, DBDs) are released at sync point and have to
be reacquired if the same IMS program continues processing the next message. Consequently, RELEASE(DEALLOCATE) is the default and is recommended.

### 15.2.2.5 BIND ISOLATION Parameter

This parameter determines when shared page locks are released:

- **At commit time**, when repeatable reads are requested with ISOLATION(RR).
  
  This is similar to accessing a DL/I segment with a Q command code in the SSA. It prevents any updaters from gaining access to the page, but it increases the number of concurrent locks held by the transaction.

- **When the position moves off the page**, if cursor stability is requested with ISOLATION(CS). This is the same as standard record locking in IMS; when the PCB position moves to a new root, the previous root is unlocked (if not updated).

Unless the application has a specific need for referenced data to remain unchanged, the recommendation is to use ISOLATION(CS).

### 15.2.2.6 BIND CURRENT DATA Parameter

DB2 V3 introduced a lock-avoidance facility that is equivalent to PROCOPT=GO in IMS. The parameter values are:

- CURRENT DATA(YES) → take locks
- CURRENT DATA(NO) → take no locks

This function provides DB2 with a dirty read capability. When using CURRENT DATA(NO) in the BIND, DB2 does not lock before a read.

In DB2, it is not always possible to specify the update intention in the EXEC SQL SELECT call. Therefore, the use of CURRENT DATA(NO) can be dangerous and can cause integrity exposures when two programs attempt to update the same data without locking it at read time.

For more details refer to the DATABASE2 V4 Administration Guide (SC26-3265).

### 15.2.3 Static versus Dynamic SQL

Static and dynamic SQL are not equal, as explained below.

#### 15.2.3.1 Static SQL

With static SQL, SQL statements are embedded within a program and prepared during the program preparation process before program execution. After the statements are prepared, they do not change, although the values of the host variables specified by the statements may change.

For best performance, use static SQL for conventional predefined transactions or programs. Static SQL avoids the EXEC SQL PREPARE overhead for access path selection.

---

26 To improve concurrency, the DB2 V4 provides a new ability to allow an application to read uncommitted data, known as read-through locks or uncommitted read (UR) isolation, but commonly called dirty read.
15.2.3.2 Dynamic SQL

With dynamic SQL, the SQL source is contained in host language variables and not coded explicitly into the application program. When the program is executed, the SQL code can be varied or dynamically created before being executed.

In a transaction processing environment, the use of dynamic SQL usually has a negative impact on performance. Dynamic SQL is used in environments where the SQL statement is defined at execution time. A typical example is the query management facility (QMF). This is not normally the sort of environment used for IMS transaction processing.

Dynamic SQL carries the PREPARE overhead. The actual SQL request is formulated at execution time, and placement markers can be used for variable substitution. The key factor is the amount of PREPARE processing relative to EXECUTE processing. For simple SQL, the PREPARE overhead can be very significant (10% or more). The PREPARE overhead includes both CPU and I/O to access the DB2 catalog and perform access path selection. In extreme cases, the DB2 catalog can be a bottleneck.

If dynamic SQL must be used, then it is best to avoid multiple PREPAREs. If the statement is going to be used only once, it is better to code EXEC SQL EXECUTE IMMEDIATE. If the statement is going to be executed several times, possibly with different values, it is better to code EXEC SQL PREPARE once and code EXEC SQL EXECUTE with placement marker substitution.

Dynamic SQL carries potential security exposures. With static SQL, only the binder requires the underlying data access privileges. With dynamic SQL, each user must have the underlying data access privileges. Giving each user explicit data privileges can lead to possible security exposures.

15.2.4 Security Controls

Security controls occur at various stages of processing, depending on the system definition and the security product being used (RACF, for example). The following security controls are frequently used:

- IMS authorization to access protected data sets
- IMS authorization to connect to DB2 subsystem
- User authorization to sign on to IMS
- User authorization to access an IMS transaction
- User authorization to sign on to DB2
- User authorization to submit IMS or DB2 commands

Security verification tasks affect performance. The overall recommendation is to use the minimum level that meets all security requirements.

15.2.5 Dispatching Priority

Figure 104 on page 337 shows the suggested dispatching priorities for the address spaces in an IMS TM environment where DB2 provides services to online as well as batch programs.
IRLM is at the top of the list. This minimizes the holding of locks. When locking services are required, they need to be processed immediately to minimize any possible locking contention implied in the applications.

Do not share the same IRLM between IMS and DB2. There is no global deadlock detection. A deadlock involving DL/I and DB2 resources shows as a time-out. It is better to have separate IRLMs for availability reasons. It is also a good idea to isolate DB2 access to specific IMS dependent regions for availability reasons.

Because most DB2 and IRLM functions run under the caller’s TCB, data access is performed under the control of the user address-space TCB (IMS dependent-region address spaces) and its related priority.

### 15.3 Tools for Monitoring

All standard products that are normally used for separate IMS and DB2 subsystems can be implemented to monitor the system when IMS and DB2 are used together. The main monitoring products are:

- **Resource Measurement Facility (RMF) II**
  
  RMF monitors the physical resource utilization of the subsystem environment.

- **IMS Performance Analysis Reporting System (IMSPARS)**
  
  The response time analysis is normally calculated, from the IMS log records.

- **IMS Monitor**
  
  The IMS monitor traces all IMS interactions with DB2, such as thread creation, signon, SQL calls, commit, and thread termination.

- **DB2 Trace Facility**
  
  Using the DB2 trace, subsystem data and events related to performance or accounting can be recorded for further use by other print programs.

- **DB2 Performance Monitor for MVS (DB2PM)**
  
  DB2PM produces reports from DB2 statistics, accounting, and performance trace data collected in the system management facility (SMF), GTF, or user files. DB2PM allows users to perform online monitoring of the DB2 system and applications through ISPF menus and panels and an interactive report facility.
• Service Level Reporter

Data coming from different sources (SMF, IMS log, RMF, and DB2) can be collected into the service-level reporter database to produce reports based on user specifications.

• IMS DISPLAY Command

This command allows the DB2 status, database status, usage of buffer pools, and locks to be observed as they stand at one point in time.

The following is an example (using a command recognition character of '-' for the DISPLAY command.

```
/SSR - DISPLAY THREAD (*).
/SSR - DISPLAY DATABASE(xxxx) SPACE(*) RESTRICT
/SSR - DISPLAY DATABASE(xxxx) SPACE(*) LOCKS
/SSR - DISPLAY BUFFERPOOL (*).
```

15.3.1 IMS Monitor

The IMS monitor provides statistics on these significant DB2 events:

- Service calls such as create thread, signon, commit Phase 1, commit Phase 2, and terminate thread.
- Normal SQL calls
- Command calls using the IMS /SSR command

The information is reported by subsystem, region or PSB. You get the number of occurrences of each type of calls and their elapsed time. Each call is one IWAIT (the NON-IWAIT time is always zero).

Examples are shown in Figures 105 to 109 on pages 339 to 343. Further information can be found in A User's Guide to the IMS Monitor (GG66-3134).
### IMS Monitor Region Summary DB2 Service and Command Calls

<table>
<thead>
<tr>
<th>Service and Command Calls</th>
<th>Elapsed Time</th>
<th>Not IWAIT Time (Elapsed-IWAIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2P</td>
<td>Occurrences</td>
<td>Total</td>
</tr>
<tr>
<td><strong>REGION 1</strong></td>
<td>45</td>
<td>68</td>
</tr>
<tr>
<td><strong>REGION 2</strong></td>
<td>76</td>
<td>393</td>
</tr>
<tr>
<td><strong>REGION 3</strong></td>
<td>78</td>
<td>495</td>
</tr>
<tr>
<td><strong>REGION 4</strong></td>
<td>79</td>
<td>2</td>
</tr>
<tr>
<td><strong>REGION 5</strong></td>
<td>81</td>
<td>398</td>
</tr>
<tr>
<td><strong>REGION 6</strong></td>
<td>84</td>
<td>587</td>
</tr>
<tr>
<td><strong>REGION 7</strong></td>
<td>85</td>
<td>94</td>
</tr>
<tr>
<td><strong>REGION 8</strong></td>
<td>86</td>
<td>12</td>
</tr>
<tr>
<td><strong>REGION 9</strong></td>
<td>87</td>
<td>30</td>
</tr>
<tr>
<td><strong>REGION 10</strong></td>
<td>88</td>
<td>844</td>
</tr>
<tr>
<td><strong>REGION 11</strong></td>
<td>89</td>
<td>14</td>
</tr>
<tr>
<td><strong>REGION 12</strong></td>
<td>90</td>
<td>195</td>
</tr>
<tr>
<td><strong>REGION 13</strong></td>
<td>92</td>
<td>188</td>
</tr>
<tr>
<td><strong>REGION 14</strong></td>
<td>94</td>
<td>280</td>
</tr>
<tr>
<td><strong>REGION 15</strong></td>
<td>96</td>
<td>158</td>
</tr>
<tr>
<td><strong>REGION 16</strong></td>
<td>97</td>
<td>693</td>
</tr>
<tr>
<td><strong>REGION 17</strong></td>
<td>98</td>
<td>2</td>
</tr>
<tr>
<td><strong>REGION 18</strong></td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td><strong>REGION 19</strong></td>
<td>101</td>
<td>88</td>
</tr>
<tr>
<td><strong>REGION 20</strong></td>
<td>102</td>
<td>313</td>
</tr>
<tr>
<td><strong>REGION 21</strong></td>
<td>103</td>
<td>319</td>
</tr>
<tr>
<td><strong>REGION 22</strong></td>
<td>104</td>
<td>337</td>
</tr>
<tr>
<td><strong>REGION 23</strong></td>
<td>105</td>
<td>8</td>
</tr>
<tr>
<td><strong>REGION 24</strong></td>
<td>106</td>
<td>195</td>
</tr>
<tr>
<td><strong>REGION 25</strong></td>
<td>107</td>
<td>16</td>
</tr>
<tr>
<td><strong>REGION 26</strong></td>
<td>110</td>
<td>279</td>
</tr>
<tr>
<td><strong>REGION 27</strong></td>
<td>111</td>
<td>40</td>
</tr>
<tr>
<td><strong>REGION 28</strong></td>
<td>112</td>
<td>4</td>
</tr>
<tr>
<td><strong>REGION 29</strong></td>
<td>115</td>
<td>594</td>
</tr>
</tbody>
</table>

**Totals:**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2P</td>
<td>10858</td>
<td>269014753</td>
<td>24775</td>
</tr>
</tbody>
</table>

*Figure 105. IMS Monitor Region Summary DB2 Service and Command Calls*
<table>
<thead>
<tr>
<th>DB2P</th>
<th>Normal Calls</th>
<th>Occurrences</th>
<th>Elapsed Time</th>
<th>Not I Wait Time (Elapsed - I Wait)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td>Mean</td>
</tr>
<tr>
<td><strong>REGION</strong> 45</td>
<td>109</td>
<td>2092068</td>
<td>19193</td>
<td>350237</td>
</tr>
<tr>
<td><strong>REGION</strong> 76</td>
<td>1647</td>
<td>13005647</td>
<td>7896</td>
<td>767487</td>
</tr>
<tr>
<td><strong>REGION</strong> 78</td>
<td>12079</td>
<td>4381257</td>
<td>3627</td>
<td>2081746</td>
</tr>
<tr>
<td><strong>REGION</strong> 79</td>
<td>2</td>
<td>75255</td>
<td>37627</td>
<td>74687</td>
</tr>
<tr>
<td><strong>REGION</strong> 81</td>
<td>1853</td>
<td>17343170</td>
<td>9359</td>
<td>1630921</td>
</tr>
<tr>
<td><strong>REGION</strong> 84</td>
<td>841</td>
<td>27677591</td>
<td>32910</td>
<td>3903304</td>
</tr>
<tr>
<td><strong>REGION</strong> 85</td>
<td>791</td>
<td>2015683</td>
<td>2548</td>
<td>156312</td>
</tr>
<tr>
<td><strong>REGION</strong> 86</td>
<td>6</td>
<td>269725</td>
<td>44954</td>
<td>114043</td>
</tr>
<tr>
<td><strong>REGION</strong> 87</td>
<td>262</td>
<td>5478409</td>
<td>20909</td>
<td>758485</td>
</tr>
<tr>
<td><strong>REGION</strong> 88</td>
<td>1532</td>
<td>23430661</td>
<td>15294</td>
<td>901634</td>
</tr>
<tr>
<td><strong>REGION</strong> 89</td>
<td>21</td>
<td>1514861</td>
<td>72136</td>
<td>190165</td>
</tr>
<tr>
<td><strong>REGION</strong> 90</td>
<td>204</td>
<td>17769197</td>
<td>87103</td>
<td>2779917</td>
</tr>
<tr>
<td><strong>REGION</strong> 92</td>
<td>202</td>
<td>17463934</td>
<td>86455</td>
<td>2878288</td>
</tr>
<tr>
<td><strong>REGION</strong> 94</td>
<td>2291</td>
<td>12311008</td>
<td>5373</td>
<td>485362</td>
</tr>
<tr>
<td><strong>REGION</strong> 96</td>
<td>270</td>
<td>25599107</td>
<td>94811</td>
<td>2669442</td>
</tr>
<tr>
<td><strong>REGION</strong> 97</td>
<td>881</td>
<td>28472239</td>
<td>32318</td>
<td>1315858</td>
</tr>
<tr>
<td><strong>REGION</strong> 98</td>
<td>2</td>
<td>128605</td>
<td>64302</td>
<td>128038</td>
</tr>
<tr>
<td><strong>REGION</strong> 100</td>
<td>206</td>
<td>7279810</td>
<td>35338</td>
<td>980344</td>
</tr>
<tr>
<td><strong>REGION</strong> 101</td>
<td>87</td>
<td>1455176</td>
<td>16726</td>
<td>168286</td>
</tr>
<tr>
<td><strong>REGION</strong> 102</td>
<td>4860</td>
<td>17455987</td>
<td>3591</td>
<td>974776</td>
</tr>
<tr>
<td><strong>REGION</strong> 103</td>
<td>727</td>
<td>5825322</td>
<td>8012</td>
<td>449525</td>
</tr>
<tr>
<td><strong>REGION</strong> 104</td>
<td>1229</td>
<td>15077402</td>
<td>12268</td>
<td>1105191</td>
</tr>
<tr>
<td><strong>REGION</strong> 105</td>
<td>4</td>
<td>48444</td>
<td>12111</td>
<td>33647</td>
</tr>
<tr>
<td><strong>REGION</strong> 106</td>
<td>346</td>
<td>28589173</td>
<td>82627</td>
<td>4110740</td>
</tr>
<tr>
<td><strong>REGION</strong> 107</td>
<td>27</td>
<td>1410199</td>
<td>52229</td>
<td>178564</td>
</tr>
<tr>
<td><strong>REGION</strong> 110</td>
<td>2754</td>
<td>18589597</td>
<td>6750</td>
<td>1027505</td>
</tr>
<tr>
<td><strong>REGION</strong> 111</td>
<td>419</td>
<td>9226607</td>
<td>22020</td>
<td>1702615</td>
</tr>
<tr>
<td><strong>REGION</strong> 112</td>
<td>12</td>
<td>657301</td>
<td>54775</td>
<td>138254</td>
</tr>
<tr>
<td><strong>REGION</strong> 115</td>
<td>2226</td>
<td>90514843</td>
<td>40662</td>
<td>1893543</td>
</tr>
</tbody>
</table>

...  

**TOTALS** | 59305 | 812420287 | 13699

_Figure 106. IMS Monitor Region Summary DB2 Normal Calls_
<table>
<thead>
<tr>
<th>REGION</th>
<th>OCCURRENCES</th>
<th>IWAIT TIME</th>
<th>FUNCTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3923</td>
<td>32209</td>
<td>DD=FADC9310</td>
<td>VBH</td>
</tr>
<tr>
<td>9</td>
<td>180727</td>
<td>28449</td>
<td>DD=PLDD2609</td>
<td>VBH</td>
</tr>
<tr>
<td>5</td>
<td>279319</td>
<td>117394</td>
<td>DD=PLDD2509</td>
<td>VBH</td>
</tr>
<tr>
<td>3</td>
<td>68801</td>
<td>30613</td>
<td>DD=FADC9409</td>
<td>VBH</td>
</tr>
<tr>
<td>1</td>
<td>80900</td>
<td>80900</td>
<td>DD=CMDB85PR</td>
<td>VBH</td>
</tr>
<tr>
<td>2</td>
<td>37651</td>
<td>34770</td>
<td>DD=FADC9309</td>
<td>VBH</td>
</tr>
<tr>
<td>13</td>
<td>252695</td>
<td>81961</td>
<td>DD=PLDD2604</td>
<td>VBH</td>
</tr>
<tr>
<td>3</td>
<td>45094</td>
<td>24355</td>
<td>DD=PLDD2504</td>
<td>VBH</td>
</tr>
<tr>
<td>2</td>
<td>57547</td>
<td>32374</td>
<td>DD=CIDA5715</td>
<td>VBH</td>
</tr>
<tr>
<td>3</td>
<td>66133</td>
<td>41605</td>
<td>DD=PLDD02</td>
<td>VBH</td>
</tr>
<tr>
<td>2</td>
<td>69138</td>
<td>49643</td>
<td>DD=CIDA5625</td>
<td>VBH</td>
</tr>
<tr>
<td>4</td>
<td>66837</td>
<td>30219</td>
<td>DD=CIDA5525</td>
<td>VBH</td>
</tr>
<tr>
<td>1</td>
<td>42138</td>
<td>42138</td>
<td>DD=PLDD120V</td>
<td>VBH</td>
</tr>
<tr>
<td>2</td>
<td>100823</td>
<td>76621</td>
<td>DD=PLDD50</td>
<td>VBH</td>
</tr>
<tr>
<td>..TOTAL..</td>
<td>457</td>
<td>15579174</td>
<td>34090</td>
<td></td>
</tr>
</tbody>
</table>

**DB2P CALLS**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>33741</td>
<td>1984</td>
<td>4626</td>
<td>D50</td>
</tr>
<tr>
<td>17</td>
<td>8410</td>
<td>494</td>
<td>670</td>
<td>P10</td>
</tr>
<tr>
<td>109</td>
<td>209268</td>
<td>19193</td>
<td>350237</td>
<td>PRO</td>
</tr>
<tr>
<td>17</td>
<td>7949</td>
<td>467</td>
<td>541</td>
<td>CT0</td>
</tr>
<tr>
<td>17</td>
<td>5568</td>
<td>327</td>
<td>3266</td>
<td>S00</td>
</tr>
<tr>
<td>..TOTAL..</td>
<td>177</td>
<td>2147736</td>
<td>12134</td>
<td></td>
</tr>
</tbody>
</table>

Figure 107. IMS Monitor Region IWAIT (Showing DB2 IWAITs)
<table>
<thead>
<tr>
<th>PSBNAME</th>
<th>PCB NAME</th>
<th>IWAITS</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DDN/FUNC</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQX008</td>
<td>CIDA5622</td>
<td>12</td>
<td>304322</td>
<td>25360</td>
<td>33447</td>
<td>CIDA5622</td>
<td>VBH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>329072</td>
<td>16453</td>
<td>32620</td>
<td>CIDA5522</td>
<td>VBH</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>32</td>
<td>633394</td>
<td>19793</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMDC84</td>
<td></td>
<td>7</td>
<td>192598</td>
<td>27514</td>
<td>55797</td>
<td>CMDC84</td>
<td>VBH</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>7</td>
<td>192598</td>
<td>27514</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DODA28</td>
<td></td>
<td>2</td>
<td>62836</td>
<td>31418</td>
<td>33390</td>
<td>DODA28</td>
<td>VBH</td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>2</td>
<td>62836</td>
<td>31418</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2P</td>
<td>41</td>
<td>29159</td>
<td>711</td>
<td>1747</td>
<td></td>
<td>COMMIT PH.1 P10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>149</td>
<td>9074336</td>
<td>60901</td>
<td>273177</td>
<td>NORMAL CALL PROC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>142083</td>
<td>3465</td>
<td>41155</td>
<td>SIGNON SOO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSYS TOTAL</td>
<td>231</td>
<td>9245578</td>
<td>40024</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td>5948</td>
<td>167985042</td>
<td>28242</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 108. IMS Monitor Region Summary (Showing DB2 IWAITs)
### IMS Monitor Call Summary

<table>
<thead>
<tr>
<th>PSB Name</th>
<th>PCB Name</th>
<th>Call</th>
<th>Lev</th>
<th>Stat</th>
<th>Func</th>
<th>No. Segment</th>
<th>Code</th>
<th>Calls</th>
<th>Iwaits/Call</th>
<th>Elapsed Time</th>
<th>Not Iwait Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBX053</td>
<td>I/O PCB</td>
<td>INIT</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0.00</td>
<td>122</td>
<td>1028</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(GU)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0.00</td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASRT</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18</td>
<td>0.00</td>
<td>1732</td>
<td>5411</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GU</td>
<td>0</td>
<td></td>
<td></td>
<td>QC</td>
<td></td>
<td>18</td>
<td>0.94</td>
<td>389319</td>
<td>896112</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PURG</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0.10</td>
<td>9330</td>
<td>174480</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISRT</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0.00</td>
<td>92</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHNG</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0.00</td>
<td>13104</td>
<td>209958</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GU</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1.00</td>
<td>415353</td>
<td>541528</td>
</tr>
<tr>
<td>I/O PCB</td>
<td></td>
<td>Subtotal</td>
<td>133</td>
<td>21</td>
<td>0.15</td>
<td>62408</td>
<td>57574</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIDA58</td>
<td>REPL (01)</td>
<td>TISA58</td>
<td>19</td>
<td>0</td>
<td>0.00</td>
<td>279</td>
<td>345</td>
<td>279</td>
<td>345</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GU (01)</td>
<td>TISA58</td>
<td>19</td>
<td>2</td>
<td>0.10</td>
<td>1684</td>
<td>26219</td>
<td>337</td>
<td>686</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL/I PCB</td>
<td>Subtotal</td>
<td></td>
<td>38</td>
<td>2</td>
<td>0.05</td>
<td>981</td>
<td>308</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSDA21</td>
<td>REPL (01)</td>
<td>MSSA21</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>19</td>
<td>31</td>
<td>19</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GU (01)</td>
<td>MSSA21</td>
<td>20</td>
<td>0</td>
<td>0.00</td>
<td>24</td>
<td>54</td>
<td>24</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL/I PCB</td>
<td>Subtotal</td>
<td></td>
<td>40</td>
<td>0</td>
<td>0.00</td>
<td>21</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2P</td>
<td>CREATE THRD</td>
<td>OK</td>
<td>19</td>
<td>19</td>
<td>1.00</td>
<td>454</td>
<td>508</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SIGNON</td>
<td>OK</td>
<td>21</td>
<td>21</td>
<td>1.00</td>
<td>167</td>
<td>546</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TERM THRD</td>
<td>OK</td>
<td>18</td>
<td>18</td>
<td>1.00</td>
<td>1659</td>
<td>5347</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMMIT PH.2</td>
<td>OK</td>
<td>20</td>
<td>20</td>
<td>1.00</td>
<td>164213</td>
<td>519310</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMMIT PH.1</td>
<td>OK</td>
<td>20</td>
<td>20</td>
<td>1.00</td>
<td>164552</td>
<td>437868</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NORMAL CALL</td>
<td>OK</td>
<td>60</td>
<td>60</td>
<td>1.00</td>
<td>70003</td>
<td>527754</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSYS SUBTOTAL</td>
<td></td>
<td>158</td>
<td>158</td>
<td>1.00</td>
<td>68465</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>369</td>
<td>181</td>
<td>0.49</td>
<td>51913</td>
<td>20786</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 109. IMS Monitor Call Summary (Showing DB2 Calls)
15.3.2 DB2 Performance Monitor

15.3.2.1 DB2PM Online Monitor
The DB2 performance monitor for MVS (DB2PM) Version 4 online monitor provides the capability to view an active DB2 subsystem and identify performance problems online. It displays DB2 performance information in a comprehensive format that is easy to understand and analyze. It can be used to:

- Determine DB2 performance and efficiency
- Measure an application’s performance and resource use
- Evaluate an application’s impact on other applications and the system
- Identify potential problems
- Determine tuning requirements for DB2

Refer to the *DB2 PM Online Monitor User's Guide (SB12-6165)* for more information about this tool.

15.3.2.2 DB2PM Batch Reporter
DB2PM can produce two groups of reports:

- Reports where information from several records is summarized into one entry.
- Traces where no data reduction takes place. The events are shown in time sequence as they appear in the trace file.

For both groups, reports and traces, it is possible to produce the output in various levels of detail. A problem area can be investigated in depth because:

- A problem area can be identified from a report summary.
- A problem area can be further investigated, using a report detail.
- A problem can be better analyzed using the appropriate trace.

The reports can be divided into three main types:

- Systems-oriented reports

  Provide performance information on the system and include the following:
  
  - Statistics (based on the statistics trace)
    - Number of SQL statements
    - Database buffer pool
    - EDM pool
    - Locking
    - CPU time spent in the two DB2 address spaces and the IRLM
    - Log manager
    - Subsystem services
  
  - I/O activity (based on the performance trace)
    - Active log
    - Archive log
- Database buffer pool
- EDM pool

- Workload-oriented reports can be used for:
  - Accounting
  - I/O activity by application plan
  - SQL trace (to know how an SQL statement is executed and the resources used)
  - Locking (to receive information on lock suspensions and their causes)
  - Record trace (with detailed information about DB2 events and their sequence)

- Graphics reports can be produced for:
  - Statistics
  - Accounting by field identifier
  - Accounting by DB2 identifier
  - Data distribution

In addition, auxiliary reports are always provided with the requested report, and contain:
  - System initialization parameters
  - Type of trace
  - Number and percentages of the trace records analyzed
Appendix A. Use of IMSPARS in Performance Analysis

This appendix describes some of the IMSPARS reports that are useful in analyzing the performance of an IMS system.

A.1 IMSPARS Reports

Not all of the reports generated by IMSPARS are described. Refer to the IMSPARS Description and Operations (SB21-2140) for a detailed description of all reports.

Consider the following restrictions and cautions when using IMSPARS:

- **Nonrecoverable Transactions**
  These transactions are processed only by the DC queue manager trace and the message queue utilization reports. They are also included in the CPU usage and IRUR reports.

- **MSC Transactions**
  These transactions are supported only system by system. If these transactions pass through a system, they are treated as message switches. Multisystem transit time and merged MSC logs are not supported.

- **BMP Transactions**
  BMP transactions are included in all reports, but use care when including them in the transit reports. Since BMP transactions usually execute over a longer time than MPPs, the transit time results are inflated and therefore cannot not be considered an accurate performance-tuning aid.

A.2 Transit Time Analysis Report

The transit time analysis report (Figure 110 on page 349) depicts response time by different areas of IMS processing. The report shows input queue, program switch, program execution, output queue, and total transit time for the transaction selected.

From this report, you can detect a possible bottleneck in one or more of the major processing areas. If, for example, transactions are spending a large amount of time on the input queue, the number of regions could be increased, class scheduling could be analyzed and changes made to the classes assigned to the transaction, the processing limit count of the transactions on the input queue could be increased, or the processing limit count of the transactions that are monopolizing the regions could be decreased.

A.3 Transaction Transit Log

The transaction transit log report (Figure 111 on page 350) is a detailed list of transactions processed by the IMS system. The list is sequenced by the output message time (GU to the message queue to get the message to send, log record type x ‘31’).
The report is very useful in determining whether scheduling conflicts in transaction priority or class are occurring. If input queue time is high, a further investigation of which region the transaction executed in, and when that region became idle, can be determined from the report.

A.4 Region Histogram

The region histogram (Figure 112 on page 351) is a graphical display of region activity and shows whether the region is active or idle and the transactions that were scheduled in the regions.

The report can be used to determine whether some regions are heavily utilized while some others are relatively idle during the same time. Also, transactions that are waiting for other transactions to complete can be discovered from this report.

If any checkpoints were taken during the time interval for which the histogram was selected, they are displayed on the report. At these checkpoint times, it can be noted that either a region did not schedule or terminate, or that the active regions experienced elongated elapsed execution times.

This report, along with the transaction transit log report, is useful in monitoring the flow of work through the IMS system. If long input queue time on the transit log is shown, the histogram can be run to determine which other work was executing at that same time.
<table>
<thead>
<tr>
<th>TRANSACTION CODE</th>
<th>NUMBER OF TRANSACTIONS</th>
<th>MINIMUM SWITCH TIME (SEC)</th>
<th>MINIMUM PROGRAM OUTPUT TIME (SEC)</th>
<th>MINIMUM EXECUTION TIME (SEC)</th>
<th>TOTAL SWITCH TIME (SEC)</th>
<th>TOTAL PROGRAM OUTPUT TIME (SEC)</th>
<th>TOTAL EXECUTION TIME (SEC)</th>
<th>PEAK SWITCH TIME (SEC)</th>
<th>PEAK PROGRAM OUTPUT TIME (SEC)</th>
<th>PEAK EXECUTION TIME (SEC)</th>
<th>MAXIMUM CPU TIME (MSECS)</th>
<th>AVERAGE CPU TIME (MSECS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFUB02GA CFUB02G</td>
<td>67</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL CFUB02GA</td>
<td>67</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>CFUI01GA CFUI01G</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL CFUI01GA</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>CFUI15GA CFUI10G</td>
<td>15</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL CFUI15GA</td>
<td>15</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>CFUI50GA CFUI40G</td>
<td>11</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL CFUI50GA</td>
<td>11</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>CFUU10GA CFUU10G</td>
<td>11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL CFUU10GA</td>
<td>11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>CFUU40GA CFUU40G</td>
<td>1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>COMSEC CFUU10G</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>ECTI10G</td>
<td>2</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>ECTM016</td>
<td>3</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>ECTR01G</td>
<td>1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>EMCCMP</td>
<td>1</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>EMCDMP</td>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>LBJ521</td>
<td>2</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>RPDRLNKI</td>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>RPDRLNK6</td>
<td>16</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>TOTAL COMSEC</td>
<td>45</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>ECTB01GA ECTB01G</td>
<td>1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>RWBRSMP</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TOTAL ECTB01GA</td>
<td>1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>
### IMS LOG TAPE REPORTS (V1M8)

#### TRANSACTION TRANSIT LOG

**LOG** 7/02/96 11:32:58.60

**REPORT** 14:30 7/02/96 PAGE 1

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>TRANSACTION</th>
<th>PSB</th>
<th>E</th>
<th>L</th>
<th>R</th>
<th>NO DB</th>
<th>CODE</th>
<th>CALLS</th>
<th>LTERM</th>
<th>SOURCE</th>
<th>TRANSACT</th>
<th>SOURCE</th>
<th>NO DB</th>
<th>LTERM</th>
<th>MESSAGE</th>
<th>MESSAGE</th>
<th>OUTPUT</th>
<th>SOURCE</th>
<th>OUTPUT</th>
<th>PROCESSING</th>
<th>PROCESSING</th>
<th>MESSAGE</th>
<th>MESSAGE</th>
<th>OUTPUT</th>
<th>OUTPUT</th>
<th>ELAPSED TIME (SEC.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MPS3000 RAASS11</td>
<td>RPDLNKI</td>
<td>34</td>
<td>5</td>
<td>5</td>
<td></td>
<td>- 11.36.39.7 11.36.39.7 11.36.39.7 11.36.39.7 11.36.39.7</td>
<td>1MPS3000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2MPS2614 RAAT50</td>
<td>RPDLNKH</td>
<td>5916</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.39.9 11.36.39.9 11.36.40.1 11.36.40.1 11.36.40.1</td>
<td>1MPS2614</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3MPS2194 QP2X10S</td>
<td>QP2B10S</td>
<td>2</td>
<td>2</td>
<td>E</td>
<td></td>
<td>- 11.36.40.3 11.36.40.3 11.36.40.7 11.36.40.7 11.36.40.7</td>
<td>1MPS2194</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4MPS2988 ECTM01GA</td>
<td>ECTM01G</td>
<td>45</td>
<td>2</td>
<td>3</td>
<td></td>
<td>- 11.36.40.6 11.36.40.6 11.36.40.8 11.36.40.8 11.36.40.8</td>
<td>1MPS2988</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5MPS2846 STS03</td>
<td>SP503</td>
<td>5316</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.2 11.36.41.2 11.36.41.2 11.36.41.2</td>
<td>1MPS2846</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6UAP04 UAP04TP</td>
<td>UAP04TP</td>
<td>17</td>
<td>2</td>
<td>5</td>
<td></td>
<td>- 11.36.41.6 11.36.41.6 11.36.41.6 11.36.41.6</td>
<td>1MPS2901</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7MPS2901 RAASCR1</td>
<td>RPDLNKI</td>
<td>5916</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.2 11.36.41.2 11.36.41.2 11.36.41.2</td>
<td>1MPS2901</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8MPS0993 ECT715</td>
<td>ECT715B</td>
<td>5416</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.2 11.36.41.2 11.36.41.2 11.36.41.2</td>
<td>1MPS0993</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9MPS0747 COMSEC</td>
<td>SP501</td>
<td>5716</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.6 11.36.41.6 11.36.41.6 11.36.41.6</td>
<td>1MPS0747</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10MPS1172 RAASCR1</td>
<td>RPDLNKI</td>
<td>5916</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.8 11.36.41.8 11.36.41.8 11.36.41.8</td>
<td>1MPS1172</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11MPS2170 JAA08</td>
<td>JAA08AR</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td></td>
<td>- 11.36.41.7 11.36.41.7 11.36.41.7 11.36.41.7</td>
<td>1MPS2170</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12MPS2738 ECTM01GA</td>
<td>ECTM01G</td>
<td>44</td>
<td>2</td>
<td>3</td>
<td></td>
<td>- 11.36.41.7 11.36.41.7 11.36.41.7 11.36.41.7</td>
<td>1MPS2738</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13MPS2857 EMD00</td>
<td>EMDRMP</td>
<td>62</td>
<td>2</td>
<td>3</td>
<td></td>
<td>- 11.36.41.7 11.36.41.7 11.36.41.7 11.36.41.7</td>
<td>1MPS2857</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14MPS2480 RAASS11</td>
<td>RPDLNKI</td>
<td>5916</td>
<td>5</td>
<td></td>
<td></td>
<td>- 11.36.41.7 11.36.41.7 11.36.41.7 11.36.41.7</td>
<td>1MPS2480</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15MPS0993 ECT715</td>
<td>ECT715B</td>
<td>5516</td>
<td>3</td>
<td></td>
<td></td>
<td>- 11.36.41.7 11.36.41.7 11.36.41.7 11.36.41.7</td>
<td>1MPS0993</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 111.** IMSPARS Transit Log Report
<table>
<thead>
<tr>
<th>TIME</th>
<th>CKPT</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.32.58</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11.32.59</td>
<td>**</td>
<td>1</td>
</tr>
<tr>
<td>. .</td>
<td>STOP</td>
<td>1</td>
</tr>
<tr>
<td>. .</td>
<td>ECTM01GA</td>
<td>1</td>
</tr>
<tr>
<td>11.33.00</td>
<td>**</td>
<td>1</td>
</tr>
<tr>
<td>11.33.01</td>
<td>**</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 112. IMSPARS Region Histogram Report**
A.5 CPU Usage Report

The CPU usage report (Figure 113 on page 353) can be used to determine the CPU time used by the IMS system. The report shows the average CPU time used by the transaction and the average CPU time used by the control region per transaction. Also shown on this report are the average CPU times (control region and dependent regions) used by IMS for each transaction. This number can be used as input to capacity planning projects. Another column on this report that can be useful in determining whether one region is receiving more service than another is the “Elapse/CPU Ratio” column. If the ratio increases for identical transactions that are executing in different regions and the ratio changes in these different regions, investigate lower-priority regions for either page faulting or bottlenecks caused by dispatching priority.
Figure 113. IMSPARS CPU Usage Report

<table>
<thead>
<tr>
<th>REGN</th>
<th>PROGRAM</th>
<th>TRANCODE</th>
<th>OBSERV</th>
<th>CPU TIME (MSECS)</th>
<th>ELAPSE TIME (MSECS)</th>
<th>TRANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ECCI01G</td>
<td>ECCI01GA</td>
<td>17</td>
<td>3,439</td>
<td>202</td>
<td>2.15</td>
</tr>
<tr>
<td>1</td>
<td>ECCM01G</td>
<td>ECCM01GA</td>
<td>39</td>
<td>4,280</td>
<td>110</td>
<td>2.71</td>
</tr>
<tr>
<td>1</td>
<td>ECTR01G</td>
<td>ECTR01GA</td>
<td>4</td>
<td>279</td>
<td>70</td>
<td>2.15</td>
</tr>
<tr>
<td>1</td>
<td>EMBERMP</td>
<td>EMBAO0</td>
<td>2</td>
<td>59</td>
<td>30</td>
<td>5.07</td>
</tr>
<tr>
<td>1</td>
<td>EMBRMRP</td>
<td>EMBR00</td>
<td>2</td>
<td>76</td>
<td>38</td>
<td>6.62</td>
</tr>
<tr>
<td>1</td>
<td>EMBCRMP</td>
<td>EMBC00</td>
<td>13</td>
<td>688</td>
<td>53</td>
<td>3.78</td>
</tr>
<tr>
<td>1</td>
<td>EMCRRMP</td>
<td>EMCRC00</td>
<td>13</td>
<td>1,563</td>
<td>120</td>
<td>1.98</td>
</tr>
<tr>
<td>1</td>
<td>EMCDRMP</td>
<td>EMCD00</td>
<td>25</td>
<td>2,152</td>
<td>86</td>
<td>3.39</td>
</tr>
<tr>
<td>1</td>
<td>JAA01AR</td>
<td>JAA01</td>
<td>14</td>
<td>3,226</td>
<td>230</td>
<td>2.17</td>
</tr>
<tr>
<td>1</td>
<td>JAA01AR</td>
<td>JAA02</td>
<td>3</td>
<td>141</td>
<td>47</td>
<td>3.56</td>
</tr>
<tr>
<td>1</td>
<td>JAA03AR</td>
<td>JAA03</td>
<td>6</td>
<td>2,197</td>
<td>366</td>
<td>1.82</td>
</tr>
<tr>
<td>1</td>
<td>JAA05AR</td>
<td>JAA05</td>
<td>17</td>
<td>2,521</td>
<td>148</td>
<td>2.82</td>
</tr>
<tr>
<td>1</td>
<td>JAA08AR</td>
<td>JAA08</td>
<td>16</td>
<td>2,015</td>
<td>126</td>
<td>2.43</td>
</tr>
<tr>
<td>1</td>
<td>JAB00AP</td>
<td>JAB00A</td>
<td>61</td>
<td>7,448</td>
<td>122</td>
<td>2.11</td>
</tr>
<tr>
<td>1</td>
<td>JAC01SP</td>
<td>JAC01</td>
<td>5</td>
<td>299</td>
<td>60</td>
<td>4.35</td>
</tr>
<tr>
<td>1</td>
<td>Q2Z065</td>
<td>Q2Z065</td>
<td>4</td>
<td>4,851</td>
<td>1,213</td>
<td>1.26</td>
</tr>
<tr>
<td>1</td>
<td>UAP04TP</td>
<td>UAP04</td>
<td>11</td>
<td>3,771</td>
<td>343</td>
<td>2.12</td>
</tr>
<tr>
<td>1</td>
<td>UAP06TP</td>
<td>UAP06</td>
<td>3</td>
<td>595</td>
<td>198</td>
<td>1.51</td>
</tr>
<tr>
<td>1</td>
<td>UAP07TP</td>
<td>UAP07</td>
<td>11</td>
<td>404</td>
<td>37</td>
<td>4.21</td>
</tr>
<tr>
<td></td>
<td>SUBTOTAL</td>
<td></td>
<td>266</td>
<td>40,002</td>
<td>150</td>
<td>2.26</td>
</tr>
</tbody>
</table>

CPU USAGE REPORT
The Internal Resource Usage reports provide statistics for the following resources and pools:

- Message Queue Pool
- Message Format Pool
- OSAM Buffer Pool
- VSAM Buffer Pool
- DMB Pool
- PSB Pool(s)
- PSBW Pool
- DMB Work Pool
- EPCB Pool
- Control Block Pools
- Program Isolation Enqueue/Dequeue
- DL/I Calls
- Scheduling Statistics
- Latch Conflicts
- Total Transaction Rate
- Logger Statistics
- Dispatcher Statistics
- Dynamic SAP Statistics
- DFSPPOOL Statistics (CIOP, HIOP, SPAP, FPWP, EMHB, LUMP, LUMC, and AOIP)

Three of the many resource reports created by IMSPARS are described in the subsections that follow. Various sample reports, including a VSAM buffer report, are also included. For a full description of all the reports, refer to IMSPARS Description and Operations (SB21-2140).

### A.6.1 DMB Pool Statistics

Figure 114 on page 355 shows the format of the pool statistics for the majority of the pools that are formatted on the IMSPARS reports. The reports show the storage utilization of the pools in bytes allocated and maximum bytes used.

This report is better than the result of the /DIS POOL ALL command because it can be run anytime after IMS is stopped or an OLDS is archived. Monitor and adjust the maximum number of bytes used if the maximum storage used is nearing the size of the pool. The only pool for which this does not apply is the PSB pool, which should always be as full as possible.

### A.6.2 IMSPARS DASD Log Statistics

The DASD log statistics report (Figure 114 on page 355) shows the number of records logged to the OLDS and WADS. It calculates the number of records per transaction and per second. If the LOGICAL LOGGER WAIT-FOR-BUFFERS count is high (greater than 0.1 per transaction), increase the number of OLDS buffers or analyze the I/O time to write the records to determine whether there is any interference to that data set. This is discussed in more detail in 9.20, “Log Buffer Pool” on page 200.

### A.6.3 IMSPARS Program Isolation Enqueue/Dequeue Report

The program isolation enqueue/dequeue report (Figure 115 on page 356) shows the maximum number of bytes used in the program isolation pool. The storage used is not usually very large if BMPs are not running. If BMPs are executing, however, the amount of storage used can become extremely large.

Monitor the maximum number of bytes used for a period of time and compare it against the value specified in the PIMAX control region execution parameter. If the storage used is approaching the PIMAX, increase PIMAX for the next IMS restart to eliminate the possibility of U775 abends. The other values on the report show the activity to the pool, but nothing can be done to change any of the values.
### IMS Internal Resource Usage

#### DMB Pool Statistics

<table>
<thead>
<tr>
<th>Description</th>
<th>Count/Transaction/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes in DMB Pool</td>
<td>512,000</td>
</tr>
<tr>
<td>Bytes allocated at end of report</td>
<td>121,680</td>
</tr>
<tr>
<td>Maximum bytes ever used</td>
<td>121,680</td>
</tr>
</tbody>
</table>

#### Logger Statistics

<table>
<thead>
<tr>
<th>Description</th>
<th>Count/Transaction/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Logger:</td>
<td></td>
</tr>
<tr>
<td>L-Logger records written</td>
<td>549,426, 28.32, 328.68</td>
</tr>
<tr>
<td>Check write requests</td>
<td>67,514, 3.48, 40.39</td>
</tr>
<tr>
<td>Waits for writes</td>
<td>10,863, .56, 6.50</td>
</tr>
<tr>
<td>Buffer waits: CHKPT INVOKERS</td>
<td>833, .04, .50</td>
</tr>
<tr>
<td>Buffer waits: NON-CHKPT INVOKERS</td>
<td>0, .00, .00</td>
</tr>
<tr>
<td>AWE submitted on WRT</td>
<td>0, .00, .00</td>
</tr>
<tr>
<td>Physical Logger:</td>
<td></td>
</tr>
<tr>
<td>WADS EXCPVRS</td>
<td>25,425, 1.31, 15.21</td>
</tr>
<tr>
<td>2K segment writes initiated</td>
<td>80,814, 4.17, 48.35</td>
</tr>
<tr>
<td>OLDS writes initiated</td>
<td>7,107, .37, 4.25</td>
</tr>
<tr>
<td>OLDS reads initiated</td>
<td>1, .00, .00</td>
</tr>
<tr>
<td>Internal checkpoint requests</td>
<td>309, .02, .18</td>
</tr>
<tr>
<td>Accumulative WTWT wait time</td>
<td>1,042,236, 53.72, 623.50</td>
</tr>
</tbody>
</table>

**Figure 114. IMSPARS DMB Pool Report and IMSPARS DASD Log Statistics Report**
## IMS Internal Resource Usage

### Program Isolation Enqueue/Dequeue Statistics

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Interval: 27.52 (HHHH.MM.SS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum Bytes Ever Used</strong></td>
<td>991,232</td>
</tr>
<tr>
<td><strong>Calls to Search for Resource ID</strong></td>
<td>753,693 38.85 459.88</td>
</tr>
<tr>
<td><strong>Synonyms Searched</strong></td>
<td>111,601 5.75 66.76</td>
</tr>
<tr>
<td><strong>Maximum Synonyms Searched for a Call</strong></td>
<td>317</td>
</tr>
</tbody>
</table>

### DL/I Call Statistics

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Interval: 27.52 (HHHH.MM.SS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programs that Reached Termination</strong></td>
<td>27,015 .76 19.05</td>
</tr>
<tr>
<td><strong>Transactions Process by Above Programs</strong></td>
<td>35,715 1.00 25.10</td>
</tr>
<tr>
<td><strong>Database Get Unique Calls</strong></td>
<td>500,495 14.01 352.86 49.57% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Getnext Calls</strong></td>
<td>9,249 .26 6.52 6.99% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Get next Within Parent Calls</strong></td>
<td>70,577 1.98 49.76 1.56% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Get hold Unique Calls</strong></td>
<td>124,933 3.50 88.08 12.37% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Get Hold Next Calls</strong></td>
<td>15,776 .44 11.12 1.56% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Get Hold Next Within Parent Cls</strong></td>
<td>783 .02 .55 3.54% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Insert Calls</strong></td>
<td>47,294 1.32 33.34 4.68% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Delete Calls</strong></td>
<td>17,522 .49 12.35 4.68% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Replace Calls</strong></td>
<td>93,685 2.62 66.05 9.28% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Database Deq Calls</strong></td>
<td>0 .00 .00 .00% of DL/I Calls</td>
</tr>
<tr>
<td><strong>Total Number of Database Calls</strong></td>
<td>880,314 24.65 620.64 87.19% of DL/I Calls</td>
</tr>
</tbody>
</table>

- **IOPCB Get Unique Calls**: 61,941 1.73 43.67 6.13% of DL/I Calls
- **IOPCB Get Next Calls**: 0 .00 .00 .00% of DL/I Calls
- **IOPCB Insert Calls**: 35,750 1.00 25.20 3.54% of DL/I Calls
- **Number of IOPCB Purge Calls**: 9,803 .27 6.91 .97% of DL/I Calls
- **Total Number of DC Calls**: 107,494 3.01 75.79 10.65% of DL/I Calls

---

*Figure 115. IMSPARS PI Enqueue/Dequeue and DL/I Call Reports — Program Isolation Statistics*
## Appendix A. Use of IMS\textsuperscript{\textregistered} in Performance Analysis

**IMS Internal Resource Usage**

<table>
<thead>
<tr>
<th>Description</th>
<th>Count</th>
<th>Transactions/Second</th>
<th>Interval: 27.52 (HH:MM:SS)</th>
<th>Percentage of DB Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Enqueue Requests</td>
<td>5,246</td>
<td>.15</td>
<td>3.70</td>
<td>.60%</td>
</tr>
<tr>
<td>Waits on Test Enqueues</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>Test Dequeue Requests</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>User +0 Enqueue Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>Waits on User +0 Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>User +0 Dequeue Requests</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>Update Enqueue Requests</td>
<td>358,396</td>
<td>10.03</td>
<td>252.68</td>
<td>40.71%</td>
</tr>
<tr>
<td>Waits on Update Enqueue Requests</td>
<td>2,027</td>
<td>.06</td>
<td>1.43</td>
<td>.23%</td>
</tr>
<tr>
<td>Update Dequeue Requests</td>
<td>60,590</td>
<td>1.70</td>
<td>42.72</td>
<td>6.08%</td>
</tr>
<tr>
<td>Exclusive Enqueue Requests</td>
<td>21,822</td>
<td>.61</td>
<td>15.38</td>
<td>2.48%</td>
</tr>
<tr>
<td>Waits on Exclusive Enqueue Requests</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>Exclusive Dequeue Requests</td>
<td>1</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I CHNG Calls</td>
<td>21,409</td>
<td>.60</td>
<td>15.09</td>
<td>2.12%</td>
</tr>
<tr>
<td>DL/I AUTH Calls</td>
<td>488</td>
<td>.01</td>
<td>.34</td>
<td>.05%</td>
</tr>
<tr>
<td>DL/I SETO Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I APSB Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I DPSB Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I GMSG Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I CHKP Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I XRST Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I ROLB Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I ROLB Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I SETS Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I SETU Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I INIT Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I INQY Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
<tr>
<td>DL/I LOG Calls</td>
<td>0</td>
<td>.00</td>
<td>.00</td>
<td>.00%</td>
</tr>
</tbody>
</table>

**Figure 116. IMS\textsuperscript{\textregistered} PI Enqueue/Dequeue and DL/I Call Reports — DL/I Call Statistics**
### IMS Internal Resource Usage

**Enhanced VSAM Buffer Pool Statistics**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Count/Second</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARED RESOURCE POOL ID/TYPE</td>
<td>XXXX/D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIX OPTION: INDEX/BLOCK/DATA</td>
<td>N/N/N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUFFER SIZE</td>
<td>2,048</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBR OF BUFRS IN SUBPL</td>
<td>1,024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBR OF HS BUFRS FOR SUBPL</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBR OF WRT ERRS IN SUBPL</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LARGEST NBR OF WRT ERRS IN SUBPL</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBR OF RETRIEVE BY RBA CALLS</td>
<td>20</td>
<td>.01</td>
<td>99.03% OF RETRIEVE CALLS</td>
</tr>
<tr>
<td>NBR OF RETRIEVE BY KEY CALLS</td>
<td>73,148</td>
<td>51.57</td>
<td>99.97% OF RETRIEVE CALLS</td>
</tr>
<tr>
<td>TOTAL NBR OF RETRIEVE CALLS</td>
<td>73,168</td>
<td>51.58</td>
<td></td>
</tr>
<tr>
<td>NBR OF LOGICAL RCDS ISRTS/ESDS</td>
<td>0</td>
<td>.00</td>
<td>.00% OF UPDATE REQUESTS</td>
</tr>
<tr>
<td>NBR OF LOGICAL RCDS ISRTS/KSDS</td>
<td>21,434</td>
<td>15.11</td>
<td>99.78% OF UPDATE REQUESTS</td>
</tr>
<tr>
<td>NBR OF LOGICAL RCDS ALTRED IN SUBPL</td>
<td>47</td>
<td>.03</td>
<td>.22% OF UPDATE REQUESTS</td>
</tr>
<tr>
<td>TOTAL NBR OF UPDATES</td>
<td>21,481</td>
<td>15.14</td>
<td></td>
</tr>
<tr>
<td>NBR OF BACKGRND WRITE REQUESTS</td>
<td>0</td>
<td>.00</td>
<td>.00% OF CALLS TO VSAM</td>
</tr>
<tr>
<td>NBR OF SYNCH CALLS</td>
<td>35,157</td>
<td>24.79</td>
<td>25.90% OF CALLS TO VSAM</td>
</tr>
<tr>
<td>NBR OF VSAM GET CALLS</td>
<td>100,566</td>
<td>70.90</td>
<td>74.10% OF VSAM I/O OPERATIONS</td>
</tr>
<tr>
<td>NBR OF VSAM SEARCH BUFRS CALLS</td>
<td>0</td>
<td>.00</td>
<td>.00% OF CALLS TO VSAM</td>
</tr>
<tr>
<td>TOTAL NBR OF VSAM CALLS</td>
<td>135,723</td>
<td>95.69</td>
<td></td>
</tr>
<tr>
<td>NBR OF TIMES VSAM FOUND CI IN POOL</td>
<td>87,411</td>
<td>61.63</td>
<td></td>
</tr>
<tr>
<td>NBR OF TIMES VSAM READ CI FROM DASD</td>
<td>16,560</td>
<td>11.60</td>
<td>39.17% OF VSAM I/O OPERATION</td>
</tr>
<tr>
<td>NBR OF WRITES INIT BY IMS/ESA</td>
<td>25,717</td>
<td>18.13</td>
<td>60.83% OF VSAM I/O OPERATION</td>
</tr>
<tr>
<td>NBR OF WRITES INIT BY VSAM</td>
<td>0</td>
<td>.00</td>
<td>.00% OF VSAM I/O OPERATION</td>
</tr>
<tr>
<td>TOTAL VSAM I/O OPERATIONS</td>
<td>42,277</td>
<td>29.81</td>
<td></td>
</tr>
<tr>
<td>NBR OF SUCCESSFUL VSAM READS FRM HS</td>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>NBR OF SUCCESSFUL VSAM WRITES TO HS</td>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>NBR OF FAILED VSAM READS FROM HS</td>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>NBR OF FAILED VSAM WRITES TO HS</td>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>NBR OF PLH WAITS</td>
<td>0</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 117. IMSPARS VSAM Buffer Pool Statistics Report*
<table>
<thead>
<tr>
<th>INTERVAL</th>
<th>MSG LENGTH</th>
<th>MSG AVG LENGTH</th>
<th>INPUT TRANSACTION COUNT SHMSG LGMSG</th>
<th>TRANSACTION -MESSAGE SWITCH-- CNT SHMSG LGMSG</th>
<th>Transaction PROGRAM SWITCH-- COUNT SHMSG LGMSG</th>
<th>OUTPUT MESSAGE-- COUNT SHMSG LGMSG</th>
<th>-----TOTALS----- COUNT SHMSG LGMSG</th>
<th>PCT</th>
<th>PCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00260-00269</td>
<td>269</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>105 105</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>00290-00299</td>
<td>299</td>
<td>10452 10452</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>144 144</td>
<td>10596</td>
<td>10596</td>
</tr>
<tr>
<td>00300-00309</td>
<td>305</td>
<td>3412 3412</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3412</td>
<td>3412</td>
</tr>
<tr>
<td>00320-00329</td>
<td>324</td>
<td>1214 1214</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1214</td>
<td>1214</td>
</tr>
<tr>
<td>00380-00389</td>
<td>384</td>
<td>11 11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>00400-00409</td>
<td>407</td>
<td>540 540</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>540</td>
<td>540</td>
</tr>
<tr>
<td>00490-00499</td>
<td>494</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>00690-00699</td>
<td>695</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>00850-00859</td>
<td>859</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>15 15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>00870-00879</td>
<td>871</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>522 522</td>
<td>522</td>
<td>522</td>
</tr>
<tr>
<td>00890-00899</td>
<td>891</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>00910-00919</td>
<td>913</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>757 757</td>
<td>757</td>
<td>757</td>
</tr>
<tr>
<td>00920-00929</td>
<td>921</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>165 165</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>00940-00949</td>
<td>945</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>696 696</td>
<td>696</td>
<td>696</td>
</tr>
<tr>
<td>00950-00959</td>
<td>957</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>462 462</td>
<td>462</td>
<td>462</td>
</tr>
<tr>
<td>00980-00989</td>
<td>983</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13 13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>01010-01019</td>
<td>1016</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>554 554</td>
<td>554</td>
<td>554</td>
</tr>
<tr>
<td>01060-01069</td>
<td>1066</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>14 14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>01070-01079</td>
<td>1073</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>19 19</td>
<td>19</td>
<td>19</td>
</tr>
</tbody>
</table>

Figure 118. IMSPARS Message Queue Utilization Report
Appendix B. Use of IMSASAP II

The appendix is provided as a convenience to readers of this book who are also users of IMSASAP II. This appendix is a companion to 4.2.1, “Overview of Reduction Process and Worksheets” on page 69 but substitutes IMSASAP II report references, where possible, for IMS monitor (DFSUTR20) reports. Both IMSASAP II and DFSUTR20 reports are derived from the same source, so in many cases report values are identical. However, differences in some definitions and in the amount of data processed are reflected in some reported values. Refer to the IMSASAP Monitor Summary and Analysis Program (SB21-1793) for a discussion of these differences.

IMSASAP II can pass to DFSUTR20 those few records needed for the buffer pool statistics reports.

Compare the sections that follow to the DFSUTR20 sections that use the same IMS Monitor data, beginning with 4.2.1, “Overview of Reduction Process and Worksheets” on page 69.

B.1 IMS Resources

The IMS monitor records the data used to develop IMS system transaction profiles, message region workload indicators, I/O IWAIT time, and pool space utilization indicators. Some calculations are necessary to derive the data. The sections that follow explain where to locate the data in the IMSASAP II reports and how to perform the necessary calculations.

B.1.1 IMS System Transaction Profile

The IMS system transaction profile indicates the amount and the type of work that is being processed by the IMS system. The numbers are either in the report or are derived by dividing totals by the number of transactions processed or the number of seconds of the trace interval.

B.1.1.1 Trace Elapsed Time

The column on the IMSASAP II region summary report (Figure 119 on page 363) labeled “Elapsed SECS.MIL” shows the duration from the first event recorded until the last event recorded for the region or the trace. The reporting period may be the trace interval or may be a subset of the trace as specified on the ASAP command statement. In Figure 119, the reporting (trace) interval is 899.122 seconds. This is the number to use in any per-second calculation.

B.1.1.2 Number of Transactions Processed

The total number of transactions processed is found in “Totals” on the region summary report (Figure 119 on page 363) under the heading “Trans.” In the example, the number of transactions processed is 4000. The IMSASAP number may differ from that of the IMS monitor because IMSASAP includes in this count started transactions that were still active when the IMS monitor trace ended. The IMS monitor does not include these.
B.1.1.3 Number of Transactions per Second
This number is calculated by IMSASAP II by dividing the total number of transactions processed by the number of seconds in the trace interval. Using the numbers from Figure 119 on page 363, and driving 4000 transactions by 899 seconds gives a transaction rate of 4.45 per second. This value is reported in Figure 119 under the “Trans/Sec” column.

B.1.1.4 Number of Schedules
The total number of schedules is how many times during the trace interval the IMS scheduler completed the scheduling process for an application program. This number is found in the “Totals” line on the region summary report (Figure 119), under the heading “Scheds.” In this example, the number is 3427.

B.1.1.5 Number of Transactions Processed per Schedule
The number of transactions processed per schedule is reported in the region summary report and can be found in the “Totals” line under the heading “Trans/Schd.” The number in the example (Figure 119) is 1.17 and is calculated by dividing the total number of transactions processed by the number of schedules.

B.1.1.6 Number of DL/I Calls
The total number of DL/I calls is the total value reported in both the “Elapsed/Call” and “IWAITS/Call” You can find the value at the bottom of the report in Figure 123 on page 367. The total number in the example is 580,906. This value includes message primes as well as actual DL/I calls.

B.1.1.7 Number of DL/I Calls per Transaction
The number of DL/I calls per transaction can be found in the region summary report, Figure 121 on page 365, under the heading “Calls/Tran.” The number in this example is 145.2, which is the sum of message queue and database calls.
**TRANSACTION DATA**

<table>
<thead>
<tr>
<th>RGN NO.</th>
<th>SECS.MIL</th>
<th>SCHEDS</th>
<th>TRANS</th>
<th>TRANSF/SCHD</th>
<th>ELAP/SC.MIL.MIC</th>
<th>PERCTS OF REGN</th>
<th>ELAP/SC.DLI</th>
<th>SCH-DLI</th>
<th>ELAP/TRAN</th>
<th>CPU/TRN</th>
<th>PRCTS OF TRAN</th>
<th>ELAP/INWAIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>899.122</td>
<td>157</td>
<td>159</td>
<td>1.01</td>
<td>388.529</td>
<td>.78%</td>
<td>6.00%</td>
<td>93.22%</td>
<td>44.758</td>
<td>339.447</td>
<td>18.15%</td>
<td>60.37%</td>
</tr>
<tr>
<td>2</td>
<td>897.824</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
<td>897.824.021</td>
<td>.80%</td>
<td>100.00%</td>
<td>.00%</td>
<td>0.000</td>
<td>897.824</td>
<td>10.810</td>
<td>.00%</td>
</tr>
<tr>
<td>3</td>
<td>899.314</td>
<td>47</td>
<td>119</td>
<td>2.53</td>
<td>667.582</td>
<td>.85%</td>
<td>2.64%</td>
<td>96.51%</td>
<td>161.747</td>
<td>199.784</td>
<td>34.763</td>
<td>86.95%</td>
</tr>
<tr>
<td>4</td>
<td>899.464</td>
<td>353</td>
<td>355</td>
<td>3.10</td>
<td>259.756</td>
<td>1.31%</td>
<td>8.89%</td>
<td>89.80%</td>
<td>33.318</td>
<td>225.163</td>
<td>45.193</td>
<td>20.07%</td>
</tr>
<tr>
<td>5</td>
<td>890.667</td>
<td>302</td>
<td>615</td>
<td>2.04</td>
<td>331.104</td>
<td>.00%</td>
<td>11.23%</td>
<td>88.77%</td>
<td>0.000</td>
<td>162.591</td>
<td>0.016</td>
<td>.01%</td>
</tr>
<tr>
<td>6</td>
<td>896.676</td>
<td>275</td>
<td>277</td>
<td>1.00</td>
<td>432.737</td>
<td>1.10%</td>
<td>11.97%</td>
<td>86.73%</td>
<td>42.330</td>
<td>387.509</td>
<td>80.232</td>
<td>20.70%</td>
</tr>
<tr>
<td>7</td>
<td>899.728</td>
<td>336</td>
<td>342</td>
<td>1.02</td>
<td>739.759</td>
<td>2.73%</td>
<td>11.45%</td>
<td>85.82%</td>
<td>73.119</td>
<td>301.260</td>
<td>77.556</td>
<td>25.74%</td>
</tr>
<tr>
<td>8</td>
<td>333.717</td>
<td>9</td>
<td>9</td>
<td>1.00</td>
<td>17.941.110</td>
<td>7.16%</td>
<td>41.22%</td>
<td>51.62%</td>
<td>2655.553</td>
<td>15.284</td>
<td>500.142</td>
<td>3.27%</td>
</tr>
<tr>
<td>9</td>
<td>854.669</td>
<td>36</td>
<td>38</td>
<td>1.00</td>
<td>1423.449</td>
<td>.09%</td>
<td>4.57%</td>
<td>95.34%</td>
<td>61.175</td>
<td>194.955</td>
<td>15.41%</td>
<td>47.78%</td>
</tr>
<tr>
<td>10</td>
<td>872.943</td>
<td>31</td>
<td>31</td>
<td>1.00</td>
<td>1403.730</td>
<td>.23%</td>
<td>4.76%</td>
<td>95.01%</td>
<td>63.412</td>
<td>1340.318</td>
<td>352.582</td>
<td>26.31%</td>
</tr>
<tr>
<td>11</td>
<td>898.356</td>
<td>1</td>
<td>28</td>
<td>28.00</td>
<td>898.356.404</td>
<td>.00%</td>
<td>100.00%</td>
<td>.00%</td>
<td>0.000</td>
<td>32084.157</td>
<td>.358</td>
<td>.00%</td>
</tr>
<tr>
<td>12</td>
<td>896.466</td>
<td>414</td>
<td>415</td>
<td>1.00</td>
<td>382.161</td>
<td>2.51%</td>
<td>15.14%</td>
<td>82.35%</td>
<td>54.353</td>
<td>327.038</td>
<td>75.726</td>
<td>23.16%</td>
</tr>
<tr>
<td>13</td>
<td>896.820</td>
<td>349</td>
<td>350</td>
<td>1.00</td>
<td>295.052</td>
<td>1.58%</td>
<td>9.91%</td>
<td>88.51%</td>
<td>40.521</td>
<td>253.804</td>
<td>53.370</td>
<td>21.03%</td>
</tr>
<tr>
<td>14</td>
<td>856.342</td>
<td>28</td>
<td>28</td>
<td>1.00</td>
<td>2473.562</td>
<td>2.05%</td>
<td>7.89%</td>
<td>91.91%</td>
<td>60.809</td>
<td>2412.753</td>
<td>546.215</td>
<td>22.64%</td>
</tr>
<tr>
<td>15</td>
<td>838.317</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
<td>838.317.680</td>
<td>.00%</td>
<td>100.00%</td>
<td>.00%</td>
<td>0.000</td>
<td>838.317</td>
<td>10.810</td>
<td>.00%</td>
</tr>
<tr>
<td>16</td>
<td>327.251</td>
<td>11</td>
<td>11</td>
<td>1.00</td>
<td>17538.458</td>
<td>13.17%</td>
<td>45.78%</td>
<td>41.05%</td>
<td>3917.446</td>
<td>13621.012</td>
<td>2205.214</td>
<td>16.19%</td>
</tr>
<tr>
<td>17</td>
<td>899.181</td>
<td>114</td>
<td>115</td>
<td>1.01</td>
<td>1029.614</td>
<td>1.06%</td>
<td>12.00%</td>
<td>86.94%</td>
<td>83.456</td>
<td>937.931</td>
<td>9.923</td>
<td>1.06%</td>
</tr>
<tr>
<td>18</td>
<td>899.665</td>
<td>5</td>
<td>12</td>
<td>2.40</td>
<td>112.619.541</td>
<td>1.00%</td>
<td>61.50%</td>
<td>37.41%</td>
<td>1965.169</td>
<td>46105.988</td>
<td>22.247</td>
<td>.05%</td>
</tr>
<tr>
<td>19</td>
<td>898.369</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
<td>898.369.163</td>
<td>.00%</td>
<td>100.00%</td>
<td>.00%</td>
<td>0.000</td>
<td>898.369</td>
<td>10.010</td>
<td>.00%</td>
</tr>
<tr>
<td>20</td>
<td>877.944</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
<td>877.944.723</td>
<td>.00%</td>
<td>100.00%</td>
<td>.00%</td>
<td>0.000</td>
<td>877.944</td>
<td>10.010</td>
<td>.00%</td>
</tr>
<tr>
<td>21</td>
<td>899.419</td>
<td>216</td>
<td>221</td>
<td>1.02</td>
<td>906.928</td>
<td>1.19%</td>
<td>20.59%</td>
<td>78.22%</td>
<td>49.390</td>
<td>838.137</td>
<td>200.803</td>
<td>23.96%</td>
</tr>
<tr>
<td>22</td>
<td>898.247</td>
<td>76</td>
<td>202</td>
<td>2.66</td>
<td>767.999</td>
<td>1.50%</td>
<td>5.00%</td>
<td>93.50%</td>
<td>176.821</td>
<td>222.401</td>
<td>49.922</td>
<td>22.45%</td>
</tr>
<tr>
<td>23</td>
<td>895.321</td>
<td>29</td>
<td>29</td>
<td>1.00</td>
<td>1068.111</td>
<td>.12%</td>
<td>3.48%</td>
<td>96.40%</td>
<td>35.625</td>
<td>1032.487</td>
<td>169.249</td>
<td>16.39%</td>
</tr>
<tr>
<td>24</td>
<td>899.216</td>
<td>307</td>
<td>312</td>
<td>1.02</td>
<td>356.774</td>
<td>3.16%</td>
<td>9.02%</td>
<td>87.82%</td>
<td>92.571</td>
<td>259.969</td>
<td>54.508</td>
<td>20.97%</td>
</tr>
<tr>
<td>25</td>
<td>898.916</td>
<td>335</td>
<td>338</td>
<td>1.01</td>
<td>342.694</td>
<td>2.66%</td>
<td>10.11%</td>
<td>87.23%</td>
<td>71.428</td>
<td>268.858</td>
<td>73.341</td>
<td>27.28%</td>
</tr>
<tr>
<td><strong>TOTL</strong></td>
<td><strong>899.769</strong></td>
<td><strong>3,427</strong></td>
<td><strong>4,000</strong></td>
<td><strong>1.17</strong></td>
<td><strong>2,016.262</strong></td>
<td><strong>1.27%</strong></td>
<td><strong>31.48%</strong></td>
<td><strong>67.25%</strong></td>
<td><strong>78.067</strong></td>
<td><strong>1,669.548</strong></td>
<td><strong>73.116</strong></td>
<td><strong>4.40%</strong></td>
</tr>
</tbody>
</table>

Figure 119. IMSASAP II Region Summary — Transaction
### **REGIONS SUMMARY**

<table>
<thead>
<tr>
<th>RGN NO.</th>
<th>RGN TRANS</th>
<th>CALL</th>
<th>IHTS</th>
<th>SCHD ON Q</th>
<th>NOMSG</th>
<th>BLDR I/C</th>
<th>DBAS</th>
<th>NTIW</th>
<th>NO. AVG ELAP</th>
<th>BLK AVG ELAP</th>
<th>BUSY AVG ELAP</th>
<th>SCHED NOTIWT AVG ELAP</th>
<th>SYNCH DB AVG ELAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.18</td>
<td>4.30</td>
<td>1.38</td>
<td>0.00</td>
<td>99.4%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.60</td>
<td>0.24</td>
<td>0.00</td>
<td>99.9%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>3</td>
<td>0.13</td>
<td>3.51</td>
<td>0.76</td>
<td>1.47</td>
<td>99.8%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>4</td>
<td>0.39</td>
<td>7.91</td>
<td>3.18</td>
<td>0.01</td>
<td>99.4%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>5</td>
<td>0.69</td>
<td>6.63</td>
<td>3.56</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>6</td>
<td>0.31</td>
<td>10.13</td>
<td>3.01</td>
<td>0.00</td>
<td>98.7%</td>
<td>.1%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>7</td>
<td>0.38</td>
<td>12.18</td>
<td>2.86</td>
<td>0.00</td>
<td>99.4%</td>
<td>.2%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>8</td>
<td>0.03</td>
<td>3.62</td>
<td>1.18</td>
<td>0.00</td>
<td>99.9%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>9</td>
<td>0.04</td>
<td>4.70</td>
<td>1.10</td>
<td>0.00</td>
<td>99.8%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>10</td>
<td>0.03</td>
<td>6.79</td>
<td>3.56</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>11</td>
<td>0.46</td>
<td>11.94</td>
<td>3.94</td>
<td>0.00</td>
<td>99.2%</td>
<td>.2%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>12</td>
<td>0.39</td>
<td>8.93</td>
<td>3.87</td>
<td>0.00</td>
<td>99.6%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>13</td>
<td>0.03</td>
<td>6.03</td>
<td>1.44</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>14</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>15</td>
<td>0.03</td>
<td>46.91</td>
<td>5.85</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>16</td>
<td>0.13</td>
<td>7.58</td>
<td>4.46</td>
<td>0.00</td>
<td>99.6%</td>
<td>.1%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>17</td>
<td>0.01</td>
<td>143.54</td>
<td>17.82</td>
<td>0.00</td>
<td>99.6%</td>
<td>.1%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>18</td>
<td>0.00</td>
<td>348.71</td>
<td>3.86</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>19</td>
<td>0.00</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
<td>99.9%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>20</td>
<td>0.25</td>
<td>18.34</td>
<td>3.60</td>
<td>0.00</td>
<td>99.8%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>21</td>
<td>0.22</td>
<td>6.72</td>
<td>1.35</td>
<td>1.17</td>
<td>99.6%</td>
<td>.2%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>22</td>
<td>0.03</td>
<td>1.74</td>
<td>0.49</td>
<td>0.00</td>
<td>99.9%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>23</td>
<td>0.35</td>
<td>5.72</td>
<td>1.89</td>
<td>0.01</td>
<td>98.9%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
<tr>
<td>24</td>
<td>0.38</td>
<td>10.24</td>
<td>2.58</td>
<td>0.00</td>
<td>99.6%</td>
<td>.2%</td>
<td>.1%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>.0%</td>
<td>156</td>
</tr>
</tbody>
</table>

**TOTAL** 4.45 645.62 67.89 0.05 95.0% 0.1% 0.2% 0.0% 0.2% 3332 4.045.442 318 42.733 3403 9.224 0

Figure 120. IMSASAP II Region Summary — Scheduling
**CALL DATA**

<table>
<thead>
<tr>
<th>RGN NO.</th>
<th>TRAN</th>
<th>INTS</th>
<th>CALLS</th>
<th>MESS</th>
<th>QUEUE</th>
<th>CALLS</th>
<th>DATABASE CALLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/TRAN</td>
<td>/TRAN</td>
<td>/CALL</td>
<td>/TRAN</td>
<td>/TRAN</td>
<td>/CALL</td>
<td>/TRAN /TRAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SC.MIL.MIC CPY DLA INT</td>
</tr>
</tbody>
</table>

**MESSAGE QUEUE CALLS**

<table>
<thead>
<tr>
<th>DATA CALLS</th>
<th>PCT OF CALL</th>
<th>ELAP/CALL</th>
<th>CALLS</th>
<th>INTS</th>
<th>ELAP/CALL</th>
<th>PCT OF CALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>/TRAN /TRAN /CALL</td>
<td>/TRAN /TRAN /CALL</td>
<td>SC.MIL.MIC CPY DLA INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Region Summary**

| Region | Calls | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % | % |
| 1      | 24.3 | 7.8 | 0.32 | 7.2 | 1.8 | 0.25 | 7.561 | 2.1% | 97.9% | 27.3% | 17.1 | 6.0 | 0.35 | 8.794 | 3.9% | 96.1% | 71.2% |
| 2      | 543.0 | 218.0 | 0.40 | 82.0 | 0.0 | 0.00 | 73.218 | 1.1% | 99.9% | 0.0% | 461.0 | 218.0 | 0.47 | 16.496 | 43.7% | 56.3% | 47.2% |
| 3      | 26.5 | 5.7 | 0.22 | 16.1 | 1.9 | 0.12 | 5.131 | 3.4% | 96.6% | 13.7% | 10.4 | 3.9 | 0.37 | 8.741 | 4.9% | 95.1% | 70.0% |
| 4      | 20.0 | 8.0 | 0.40 | 7.2 | 1.2 | 0.16 | 5.417 | 1.6% | 98.4% | 28.3% | 12.8 | 6.9 | 0.54 | 11.136 | 3.4% | 96.6% | 67.0% |
| 5      | 9.6 | 5.2 | 0.54 | 4.5 | 1.9 | 0.42 | 8.318 | 2.5% | 97.5% | 39.1% | 5.1 | 3.3 | 0.64 | 14.081 | 1.9% | 98.1% | 81.9% |
| 6      | 32.8 | 9.8 | 0.30 | 7.2 | 0.5 | 0.07 | 3.540 | 3.7% | 96.3% | 18.1% | 25.6 | 9.2 | 0.36 | 10.218 | 3.0% | 97.0% | 77.1% |
| 7      | 32.0 | 7.5 | 0.23 | 6.2 | 1.7 | 0.28 | 9.920 | 1.1% | 98.9% | 31.8% | 25.8 | 5.8 | 0.22 | 6.011 | 5.3% | 94.7% | 72.1% |
| 8      | 134.3 | 43.9 | 0.33 | 3.3 | 2.0 | 0.60 | 59.990 | 36.4% | 63.6% | 8.6% | 131.0 | 41.9 | 0.32 | 12.581 | 21.9% | 78.1% | 49.3% |
| 9      | 67.4 | 19.2 | 0.28 | 7.6 | 3.5 | 0.45 | 15.992 | 1.2% | 98.8% | 57.9% | 59.7 | 15.7 | 0.26 | 9.113 | 4.7% | 95.3% | 71.1% |
| 10     | 113.5 | 31.1 | 0.23 | 8.0 | 4.0 | 0.50 | 13.636 | 1.4% | 98.6% | 47.0% | 124.5 | 27.1 | 0.22 | 4.631 | 6.0% | 94.0% | 65.5% |
| 11     | 217.9 | 114.3 | 0.52 | 26.3 | 17.3 | 0.66 | 17.084 | 0.6% | 99.4% | 44.1% | 191.6 | 97.0 | 0.51 | 17.179 | 35.5% | 64.5% | 54.7% |
| 12     | 25.8 | 8.5 | 0.33 | 7.0 | 1.9 | 0.27 | 8.146 | 1.6% | 98.4% | 44.6% | 18.8 | 6.6 | 0.35 | 9.600 | 3.8% | 96.2% | 74.0% |
| 13     | 22.9 | 9.9 | 0.43 | 7.4 | 1.2 | 0.16 | 5.170 | 2.1% | 97.9% | 26.6% | 15.5 | 8.7 | 0.56 | 11.318 | 3.1% | 96.9% | 70.5% |
| 14     | 184.3 | 44.0 | 0.24 | 7.5 | 3.9 | 0.52 | 13.804 | 1.3% | 98.7% | 45.6% | 176.8 | 40.1 | 0.23 | 6.135 | 5.4% | 94.6% | 65.8% |
| 15     | 22.0 | 7.0 | 0.00 | 7.0 | 0.0 | 0.00 | 11.420 | 1.8% | 98.2% | 0.0% | 0.0 | 0.0 | 0.00 | 0.000 | 0.0% | 0.0% | 0.0% |
| 16     | 1395.5 | 174.2 | 0.12 | 8.4 | 32.3 | 3.86 | 37.231 | 9.4% | 90.6% | 69.4% | 1387.1 | 141.9 | 0.10 | 6.701 | 57.0% | 43.0% | 22.9% |
| 17     | 59.3 | 34.9 | 0.59 | 6.9 | 3.4 | 0.49 | 12.118 | 1.0% | 99.0% | 30.6% | 52.4 | 31.5 | 0.60 | 8.768 | 3.8% | 96.2% | 66.0% |
| 18     | 10.8 | 1325.7 | 0.12 | 5.3 | 3.9 | 0.75 | 7.258 | 10.3% | 89.7% | 42.6% | 10.8 | 1331.8 | 0.12 | 3.716 | 37.4% | 62.6% | 27.9% |

**Total**

<table>
<thead>
<tr>
<th>RGN NO.</th>
<th>TRAN</th>
<th>INTS</th>
<th>CALLS</th>
<th>MESS</th>
<th>QUEUE</th>
<th>CALLS</th>
<th>DATABASE CALLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/TRAN</td>
<td>/TRAN</td>
<td>/CALL</td>
<td>/TRAN</td>
<td>/TRAN</td>
<td>/CALL</td>
<td>/TRAN /TRAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SC.MIL.MIC CPY DLA INT</td>
</tr>
</tbody>
</table>

**Figure 121. IMSASAP II Region Summary — Call Information**
<table>
<thead>
<tr>
<th>Region No.</th>
<th>Average SD/AVG Max-Value</th>
<th>Average SD/AVG Max-Value</th>
<th>Average SD/AVG Max-Value</th>
<th>Average SD/AVG Max-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals All Regions</td>
<td>2.016.262 17.29 898.369.163</td>
<td>388.529 1.00 2.390.016</td>
<td>897.824.021 .00 897.824.021</td>
<td>667.582 1.14 2.983.827</td>
</tr>
<tr>
<td>Region No. 1</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
<tr>
<td>Region No. 2</td>
<td>Average SD/AVG Max-Value</td>
<td>259.756 1.71 5.229.369</td>
<td>331.104 4.34 24.751.529</td>
<td>432.737 3.71 26.125.145</td>
</tr>
<tr>
<td>Region No. 3</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 4</td>
<td>Region No. 5</td>
<td>Region No. 6</td>
</tr>
<tr>
<td>Region No. 8</td>
<td>Average SD/AVG Max-Value</td>
<td>17.941.119.45 1.45 91.083.748</td>
<td>1.423.449 1.22 8.580.252</td>
<td>1.403.730 1.43 8.707.209</td>
</tr>
<tr>
<td>Region No. 9</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 10</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
<tr>
<td>Region No. 12</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 13</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
<tr>
<td>Region No. 14</td>
<td>Average SD/AVG Max-Value</td>
<td>17.538.458 1.20 65.449.995</td>
<td>1.029.614 .92 4.813.067</td>
<td>112.619.541 1.71 495.026.795</td>
</tr>
<tr>
<td>Region No. 15</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 23</td>
<td>Region No. 24</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
<tr>
<td>Region No. 26</td>
<td>Average SD/AVG Max-Value</td>
<td>887.944.723 .00 887.944.723</td>
<td>906.928 1.10 10.887.348</td>
<td>767.939 1.57 9.614.927</td>
</tr>
<tr>
<td>Region No. 27</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 28</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
<tr>
<td>Region No. 30</td>
<td>Average SD/AVG Max-Value</td>
<td>887.944.723 .00 887.944.723</td>
<td>906.928 1.10 10.887.348</td>
<td>767.939 1.57 9.614.927</td>
</tr>
<tr>
<td>Region No. 31</td>
<td>ELAPSED/SCHEDULE</td>
<td>Region No. 32</td>
<td>ELAPSED/SCHEDULE</td>
<td>ELAPSED/SCHEDULE</td>
</tr>
</tbody>
</table>

Figure 122. IMSASAP II Region Summary — Region Distribution
**Region Totals**

<table>
<thead>
<tr>
<th>Region</th>
<th>Detail</th>
<th>Calls</th>
<th>Elapsed Time</th>
<th>Avg Elapsed/Calls</th>
<th>Avg Elapsed/IWAIT</th>
<th>IWAITS/Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>1000</td>
<td>22</td>
<td>15</td>
<td>3.966</td>
<td>13.116</td>
<td>0.11</td>
</tr>
<tr>
<td>MIL</td>
<td>0.000</td>
<td>41</td>
<td>18</td>
<td>5.900</td>
<td>1.841</td>
<td>9.652</td>
</tr>
<tr>
<td>MIC</td>
<td>250.00</td>
<td>194</td>
<td>12</td>
<td>5.604.437</td>
<td>3.020.946</td>
<td>318</td>
</tr>
</tbody>
</table>

**Range Count in Totals All Regions**

<table>
<thead>
<tr>
<th>Range</th>
<th>SC</th>
<th>MIL</th>
<th>MIC</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>750,000</td>
<td>250,000</td>
<td>100,000</td>
<td>50,000</td>
</tr>
</tbody>
</table>

**Figure 123. IMSASAP II Region Summary — Region Detail**

Appendix B. Use of IMSASAP II
B.1.1.8 Number of Message Queue I/Os
The number of message queue I/Os is obtained from the total system IWAIT summary report (Figure 124 on page 370) by adding the number of I/O IWAITs for the LGMSG (2557), SHMSG (531), and QBLKS (81) data sets. In Figure 124 on page 370, the total of 3169 message queue I/Os results from adding 2557+531+81. This number can differ from that of the IMS monitor report, which includes the IWAITs for PURGE completion and the IWAITs for the DECB.

B.1.1.9 Number of Message Queue I/Os per Transaction
This value is derived by dividing the total number of message queue I/Os by the number of transactions processed. The example shows 3169/4000 = 0.8 message queue I/Os per transaction.

B.1.1.10 Number of OSAM I/Os
The total number of OSAM database I/Os is reported in the total system IWAIT summary report (Figure 124 on page 370). The example shows 2084 OSAM database I/O IWAITs. This number can differ from the IMS monitor report because the IMS monitor reports the OSAM IWAITs from the buffer pool statistics, and IMSASAP reports the actual IWAIT monitor trace records. The OSAM drives additional I/Os from the OSAM disabled interrupt exit (DIE) routine and updates the buffer pool statistics; however, an IMS monitor trace IWAIT record is not created. The IMS monitor buffer pool statistics can be run by passing the trace records to the IMS monitor report program DFSUTR20. This procedure is described in IMSASAP Monitor Summary and Analysis Program (SB21-1793).

B.1.1.11 Number of VSAM I/Os
An estimate of the total number of VSAM I/Os is reported in the total system IWAIT Summary report. The estimate is controlled by setting a threshold value for elapsed VSAM IWAIT time. Only IWAIT times exceeding the threshold are designated by IMSASAP II as I/O IWAITs. Values below the threshold are counted as non-I/O IWAITs and are excluded from calculations involving VSAM IWAITs. Only actual VSAM I/Os are reported to the IMS Monitor, so it is recommended that the default threshold of 5 ms be changed to zero ms. The number of VSAM I/Os reported in Figure 124 on page 370 is 58,996.

B.1.1.12 Number of Database I/Os per DL/I Call
The total number of database I/Os is the sum of the number of OSAM I/Os and the number of VSAM I/Os. To calculate the total number of I/Os per call, divide the total number of database I/Os by the number of DL/I calls. The numbers in the example on page 594 are (2084+58,996)/580,906 = 0.11 database I/Os. The number of calls used in this calculation is the number of database DL/I calls plus the number of data communication, calls including message prime.

B.1.1.13 Number of Format I/Os
The total number of format pool I/Os is derived from the data on the total system IWAIT summary report. It is calculated by adding the values in the “Count” column, next to “Message Formats” Using Figure 124 on page 370, add the number of I/Os for loading the formats and the number of I/Os for reading the directories. This gives the total number of format buffer pool I/Os.
B.1.1.14 Number of Format I/Os per Transaction
This number can be calculated by dividing the total number of format I/Os by the number of transactions processed, or 1071/4000=0.27.

B.1.1.15 Number of ACBLIB I/Os
The I/Os to ACBLIB, if any, are summarized in the total system IWAIT summary report. Under the “Count” column, next to “Block Loading,” the values for items PSBS, INTLIST, and DMB indicate I/Os that resulted from obtaining PSBs, intent lists, and DMBs in order to complete scheduling. Add these three values to obtain the number of ACBLIB I/Os during the monitor interval.

Figure 124 on page 370 shows only the PSB I/Os. Adding the I/Os 269+0+0 results in the total system ACBLIB I/Os, 269.

B.1.1.16 Number of ACBLIB I/Os per Schedule
The total number of ACBLIB I/Os divided by the number of schedules yields the number of ACBLIB I/Os per schedule. Using the example, divide 269 by 347 which yields 0.08 ACBLIB I/Os per schedule.
### TOTAL SYSTEM IWAIT SUMMARY

<table>
<thead>
<tr>
<th>SYSTEM DATASETS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGMSG</td>
<td>2,557</td>
<td>8.569</td>
<td>1.362</td>
<td>181.020</td>
<td>3.9%</td>
<td>2.54%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBLKS</td>
<td>81</td>
<td>16.852</td>
<td>.664</td>
<td>74.486</td>
<td>.12%</td>
<td>.16%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHMSG</td>
<td>531</td>
<td>10.087</td>
<td>1.651</td>
<td>210.089</td>
<td>.81%</td>
<td>.62%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATABASE IWAITS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSAM</td>
<td>2,084</td>
<td>20.522</td>
<td>1.249</td>
<td>306.408</td>
<td>3.18%</td>
<td>4.97%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM</td>
<td>58,996</td>
<td>12.854</td>
<td>1.871</td>
<td>3,020.946</td>
<td>89.95%</td>
<td>88.09%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSBS</td>
<td>269</td>
<td>32.232</td>
<td>1.379</td>
<td>607.110</td>
<td>.41%</td>
<td>1.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGM ISOLATN</td>
<td>34</td>
<td>60.690</td>
<td>1.557</td>
<td>412.079</td>
<td>.24%</td>
<td>.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESSAGE FORMATS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT I/O</td>
<td>536</td>
<td>25.352</td>
<td>.932</td>
<td>355.263</td>
<td>.82%</td>
<td>1.58%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORMAT I/O</td>
<td>39</td>
<td>21.248</td>
<td>.831</td>
<td>65.278</td>
<td>.27%</td>
<td>.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK I/O</td>
<td>535</td>
<td>16.630</td>
<td>1.169</td>
<td>248.027</td>
<td>.82%</td>
<td>1.03%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA2 HITS</td>
<td>53</td>
<td>0.001</td>
<td>.000</td>
<td>0.001</td>
<td>.37%</td>
<td>.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCHEDULER IWAITS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPP BLR BUSY</td>
<td>49</td>
<td>100.380</td>
<td>1.571</td>
<td>664.140</td>
<td>.34%</td>
<td>.03%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO MSG/CHKPT</td>
<td>3,031</td>
<td>4.186.462</td>
<td>2.709</td>
<td>216.147.039</td>
<td>21.34%</td>
<td>87.30%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPP INTENT</td>
<td>102</td>
<td>219.769</td>
<td>1.042</td>
<td>1,046.912</td>
<td>.72%</td>
<td>.15%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAITING/INPUT</td>
<td>301</td>
<td>2,625.395</td>
<td>1.912</td>
<td>71,922.573</td>
<td>2.12%</td>
<td>5.44%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHECKPOINTS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECKPOINTS</td>
<td>1</td>
<td>3.661.516</td>
<td>.000</td>
<td>3.661.516</td>
<td>.01%</td>
<td>.03%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LINE/NODE IWAITS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE INTERPT</td>
<td>265</td>
<td>15.607</td>
<td>14.415</td>
<td>3,667.869</td>
<td>1.87%</td>
<td>.03%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VTAM TRANSFR</td>
<td>10,384</td>
<td>98.065</td>
<td>2.095</td>
<td>4,380.657</td>
<td>73.10%</td>
<td>7.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TOTALS

<table>
<thead>
<tr>
<th>TOTALS</th>
<th>COUNT</th>
<th>SC.MIL.MIC</th>
<th>X AVG</th>
<th>SC.MIL.MIC</th>
<th>IWAIT</th>
<th>IWT</th>
<th>ELP</th>
</tr>
</thead>
<tbody>
<tr>
<td>65,589</td>
<td>13.126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14,206</td>
<td>1.023.212</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 124. IMSASAP II Total System IWAIT Summary
B.1.1.17 Number of I/Os per Transaction
IMSASAP II computes the total number of I/Os by adding the number of message queue, OSAM, VSAM, format, and ACBLIB I/Os. The sum of 65,589 is shown in the total system IWaits summary report (Figure 124 on page 370). To obtain the number of I/Os per transaction, divide the sum by the total number of transactions processed (Figure 125).

\[
\frac{3169 + 2084 + 58996 + 1071 + 269 + 0}{4000} = 16.4
\]

Figure 125. Number of I/Os Per Transaction
PGMLIB I/Os for application programs and TP line I/Os are not included in this calculation.

B.1.2 Region Performance Indicators
The region indicators show the amount of work being done in each dependent region and how efficiently the work is being processed. The region summary report (Figure 119 on page 363) contains the region indicators.

B.1.2.1 Mean Schedule NOT IWAIT Time
The Schedule NOT-IWAIT time is the amount of IMS control-region processing time used to approve the scheduling of a dependent region. It can differ from the elapsed time, as all IWAIT time is subtracted from it. In this case, the number to be checked is the average schedule NOT-IWAIT time for the overall system. It is listed in the “Totl” line under the heading “Sched NOTIWT” in the region summary (Figure 120 on page 364), where the “Total” average NOT-IWAIT time is 9.224 ms. This time differs from that of the IMS monitor Figure 126 because the IMS monitor does not include the scheduling time for the WFI transactions. If the WFI transactions are backed out, the numbers would agree.

B.1.2.2 Schedule End to First DL/I Call
Schedule-end-to-first-DL/I-call time is the elapsed time between schedule approval in the IMS control region and the first DL/I call issued by the dependent region. The functions that account for the elapsed time are dispatching priority, program load time, high-level language initialization time, page faulting, application program housekeeping, and loading of subroutines.
<table>
<thead>
<tr>
<th>PROGRAM SUMMARY</th>
<th>REGION TOTALS</th>
<th>FROM 2/02/96 11.00.30 TO 2/02/96 11.15.03</th>
<th>ELAPSED= 14 MINS 59.769.397 SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>** PSB COMPARISON **</td>
<td>*** ALL VALUES ARE PERCENTS OF TOTAL ***</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="Database Calls" /></td>
<td><img src="image2" alt="Message Queue Calls" /></td>
<td><img src="image3" alt="Calls" /></td>
<td></td>
</tr>
<tr>
<td><img src="image4" alt="Figure 126. IMSASAP II Program Summary" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B.1.2.3 Region Idle Time (Occupancy)
Region idle time, as computed by IMSASAP II, is the scheduling and termination activity plus actual idle time waiting for messages, divided by the trace elapsed time. It is a good indicator as to whether another dependent region is needed to process the IMS workload, all existing regions are needed, or IMS class scheduling should be reevaluated to balance the workload. A region idle time of 35% or more is considered to be the optimum for throughput, including dependent regions that process WFI transactions, which are considered idle during scheduler IWAITS for subqueue 6. The region idle time can be calculated by subtracting the value under “Percts of Regn Elapsed Idle” on the region summary report (Figure 119 on page 363) from 100%. Another way to calculate the region occupancy is to add the SC-DLI percentage to the ACTIVE percentage under the heading of “Percts of Regn Elapsed” in Figure 119.

B.1.3 I/O IWAIT Times
The biggest single factor affecting program elapsed time and region idle time is IMS database I/O IWAIT time. Therefore, it is important to decrease I/O IWAIT times wherever possible, as any decrease in I/O IWAIT time decreases program elapsed time and increases idle time. The IWAIT summary report (Figure 127 on page 374) lists the mean and maximum I/O IWAIT times for all databases. An IWAIT summary is also available for each dependent region. To find out whether dependent-region I/O IWAIT times are a problem, refer to the “Elap/IWAIT” column of the combined IWAIT summary. Scan the “Maximum IWAIT” times for each data set to see if any maximum times are out of line with the majority. These should be investigated.

B.1.4 IMS Pool Space
Refer to 4.2.4, “IMS Buffer Pools” on page 89 for a complete discussion of IMS pool space analysis. The various buffer pool reports and the “Reports” report referred to are obtainable when using IMSASAP II by passing a small subset of IMS monitor records to the DFSUTR20 step. Where storage pool IWAITs are discussed, these IWAITs appear in the IMSASAP II total system IWAIT summary report (Figure 124 on page 370). Because these IWAITs occur infrequently, the sample shows no such occurrences.

B.2 IMS Application/Database Resource
Significant performance improvements can be achieved by analyzing the design of the IMS databases and their use by the application programs. Changes to the databases or applications are usually long-range performance modifications but should be considered part of the overall performance evaluation. Understanding the interrelationship between the active application programs and databases can also be helpful in making overall performance decisions for the IMS system.

B.2.1 Application Programs
All transactions should be reviewed, but normally only a certain number of applications and their associated transactions account for a large percentage of the IMS workload. The percentage of the IMS workload can be viewed in total DL/I calls issued, total number of transactions processed, or total I/Os incurred by the application program. All of these are reflected in the total elapsed time of the program. For some installations, one program accounts for most of the workload; in others, a few programs together account for the majority.
<table>
<thead>
<tr>
<th>DDNAME</th>
<th>TYPE</th>
<th>IWAIT</th>
<th>SC.MIL.MIC X AVG</th>
<th>SC.MIL.MIC WAITING</th>
<th>CALLS</th>
<th>/CALL</th>
<th>CALLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCFZP1</td>
<td>VSAM</td>
<td>3</td>
<td>15.047 .347</td>
<td>19.272</td>
<td>1</td>
<td>3.00</td>
<td>.00%</td>
</tr>
<tr>
<td>ADCFZP2</td>
<td>VSAM</td>
<td>2</td>
<td>31.640 .020</td>
<td>32.286</td>
<td>1</td>
<td>2.00</td>
<td>.00%</td>
</tr>
<tr>
<td>ADCFZP4</td>
<td>VSAM</td>
<td>2</td>
<td>17.429 .097</td>
<td>19.125</td>
<td>1</td>
<td>2.00</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAICP</td>
<td>VSAM</td>
<td>7</td>
<td>17.920 .527</td>
<td>30.757</td>
<td>4</td>
<td>1.75</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAICY1</td>
<td>VSAM</td>
<td>12</td>
<td>13.790 .703</td>
<td>31.683</td>
<td>4</td>
<td>3.00</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDP</td>
<td>VSAM</td>
<td>10</td>
<td>10.012 .831</td>
<td>24.806</td>
<td>9</td>
<td>1.11</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDX</td>
<td>VSAM</td>
<td>15</td>
<td>18.388 .822</td>
<td>35.110</td>
<td>7</td>
<td>2.14</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDY1</td>
<td>VSAM</td>
<td>7</td>
<td>15.830 .733</td>
<td>32.555</td>
<td>4</td>
<td>1.75</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDY2</td>
<td>VSAM</td>
<td>15</td>
<td>13.716 .823</td>
<td>30.168</td>
<td>10</td>
<td>1.50</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDY3</td>
<td>VSAM</td>
<td>8</td>
<td>15.626 .822</td>
<td>34.536</td>
<td>4</td>
<td>2.00</td>
<td>.00%</td>
</tr>
<tr>
<td>ACEAIDY4</td>
<td>VSAM</td>
<td>11</td>
<td>11.212 1.004</td>
<td>33.967</td>
<td>5</td>
<td>2.20</td>
<td>.00%</td>
</tr>
<tr>
<td>ADF2A0UT</td>
<td>VSAM</td>
<td>716</td>
<td>17.950 .985</td>
<td>119.123</td>
<td>373</td>
<td>1.92</td>
<td>.06%</td>
</tr>
<tr>
<td>ADF2MG5</td>
<td>VSAM</td>
<td>36</td>
<td>16.560 .682</td>
<td>54.252</td>
<td>36</td>
<td>1.00</td>
<td>.01%</td>
</tr>
<tr>
<td>ADF2SM5G</td>
<td>VSAM</td>
<td>125</td>
<td>11.265 1.374</td>
<td>132.304</td>
<td>58</td>
<td>2.16</td>
<td>.01%</td>
</tr>
<tr>
<td>ADF2W0RK</td>
<td>OSAM</td>
<td>1,878</td>
<td>20.813 1.250</td>
<td>306.408</td>
<td>1,467</td>
<td>1.28</td>
<td>.25%</td>
</tr>
<tr>
<td>AWACASS</td>
<td>VSAM</td>
<td>62</td>
<td>11.894 1.045</td>
<td>57.242</td>
<td>24</td>
<td>2.58</td>
<td>.00%</td>
</tr>
<tr>
<td>AWACTLL</td>
<td>VSAM</td>
<td>49</td>
<td>11.999 1.316</td>
<td>67.213</td>
<td>41</td>
<td>1.20</td>
<td>.01%</td>
</tr>
<tr>
<td>AWACTLP</td>
<td>VSAM</td>
<td>185</td>
<td>6.669 1.427</td>
<td>61.192</td>
<td>185</td>
<td>1.00</td>
<td>.03%</td>
</tr>
<tr>
<td>AWACTY1</td>
<td>VSAM</td>
<td>27</td>
<td>9.126 1.105</td>
<td>32.083</td>
<td>21</td>
<td>1.29</td>
<td>.00%</td>
</tr>
<tr>
<td>AWACTYP</td>
<td>VSAM</td>
<td>28</td>
<td>7.328 1.227</td>
<td>32.649</td>
<td>28</td>
<td>1.00</td>
<td>.00%</td>
</tr>
<tr>
<td>AWA1LPD</td>
<td>VSAM</td>
<td>4</td>
<td>81.846 1.418</td>
<td>281.397</td>
<td>3</td>
<td>1.33</td>
<td>.00%</td>
</tr>
<tr>
<td>AWA1LDS</td>
<td>VSAM</td>
<td>3</td>
<td>23.333 1.154</td>
<td>26.498</td>
<td>1</td>
<td>3.00</td>
<td>.00%</td>
</tr>
<tr>
<td>AWA1MP</td>
<td>VSAM</td>
<td>172</td>
<td>23.489 1.716</td>
<td>165.571</td>
<td>98</td>
<td>1.76</td>
<td>.02%</td>
</tr>
<tr>
<td>AWA1MS</td>
<td>VSAM</td>
<td>73</td>
<td>14.116 1.000</td>
<td>61.875</td>
<td>29</td>
<td>2.52</td>
<td>.00%</td>
</tr>
<tr>
<td>AWALC5C</td>
<td>VSAM</td>
<td>16</td>
<td>15.412 1.854</td>
<td>34.863</td>
<td>8</td>
<td>2.00</td>
<td>.00%</td>
</tr>
<tr>
<td>AWALDP</td>
<td>VSAM</td>
<td>11</td>
<td>26.393 1.744</td>
<td>58.410</td>
<td>7</td>
<td>1.57</td>
<td>.00%</td>
</tr>
<tr>
<td>AWALDS</td>
<td>VSAM</td>
<td>13</td>
<td>17.526 1.892</td>
<td>48.854</td>
<td>5</td>
<td>2.60</td>
<td>.00%</td>
</tr>
<tr>
<td>AWAPARS</td>
<td>VSAM</td>
<td>4</td>
<td>25.262 1.698</td>
<td>50.098</td>
<td>2</td>
<td>2.00</td>
<td>.00%</td>
</tr>
<tr>
<td>AWAPRT1</td>
<td>VSAM</td>
<td>3</td>
<td>32.592 1.405</td>
<td>49.327</td>
<td>2</td>
<td>1.50</td>
<td>.00%</td>
</tr>
<tr>
<td>AWAPRT2</td>
<td>VSAM</td>
<td>2</td>
<td>27.884 1.395</td>
<td>38.906</td>
<td>2</td>
<td>1.00</td>
<td>.00%</td>
</tr>
<tr>
<td>AWAREQ0</td>
<td>VSAM</td>
<td>30</td>
<td>16.562 1.422</td>
<td>106.397</td>
<td>18</td>
<td>1.67</td>
<td>.00%</td>
</tr>
<tr>
<td>AWARP8T</td>
<td>VSAM</td>
<td>4</td>
<td>46.521 1.330</td>
<td>71.789</td>
<td>2</td>
<td>2.00</td>
<td>.00%</td>
</tr>
</tbody>
</table>

**GRAND** *TOT 182,246 5.975 2.922 2.585.235 124,761 1.46 19.00% 100.00% 100.00% 72.466%

Figure 127. IMSASAP II IWAIT Summary
B.2.2 Databases

You can ascertain which data sets are the most frequently referenced by analyzing the IWAIT summary report (Figure 127 on page 374). Refer to the “Pct Tot IWAITS” column. Starting with the highest percentage, select DDnames until the total accumulated percentage exceeds 50%. Also observe the number of I/O occurrences (IWAITs), the mean IWAIT time (“Elap/IWAIT”), and the percentage of total IWAITs for the DDname (“Pct Tot IWTS”). Add the value for total database I/O occurrences from the “Grand Total” in the IWAITs column.

B.2.3 Program Trace Report

The program trace report provides a detail trace of events associated with a transaction. A detail line is provided for each call and optionally, the IWAITs associated with that call.

The report can be one of the most useful for pinpointing performance problems in an application program or database. It displays information on types of calls issued, length of time for IWAITs associated with a call, the database referenced, and the total time for the call.
Appendix C. Worksheets

This appendix contains various worksheets that you may find useful when analyzing your system.

C.1 CPU Resource Worksheet

The CPU resource worksheet data is collected from the RMF reports. Start by selecting the time period of most interest, perhaps by referencing the SRCS report (Figure 9 on page 42) to identify a time of high activity. Look at the column headed “CPU Util” and, using the selected time period, complete the worksheet on page 378. This requires the use of the RMF Monitor II address space state data report (ASD) (Figure 10 on page 43), the RMF Monitor I CPU activity report, and the RMF Monitor II SRCS (Figure 9 on page 42).
**Date:** __________  **System:** __________

**Time:** __________  **Interval:** __________

**CPU RESOURCES:**

**General Data:**

- _________ Average CPU Utilization
- _________ Was CPU Ever Overcommitted (Y/N)?

**Dispatching Priority:**

<table>
<thead>
<tr>
<th>DPRTY</th>
<th>JOBNAME</th>
<th>SWAPPABLE</th>
<th>DELTA CPU TIME 1</th>
<th>DELTA CPU TIME 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 128. CPU Resource Worksheet*
C.2 I/O Resource Worksheet

The utilization of the system channels and DASDs is a major concern in the area of I/O resources. Use Figure 129 on page 380 to analyze the RMF reports for I/O resources utilization. Other I/O resource utilization indicators are obtained from IMS monitor and are included in later worksheets.
### I/O RESOURCES:

**IMS DASD Volumes**

<table>
<thead>
<tr>
<th>VOLUME</th>
<th>ADDRESS</th>
<th>PVT/SHR?</th>
<th>VOLUME</th>
<th>ADDRESS</th>
<th>PVT/SHR?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Channels — Greater than 50% Utilization:**

<table>
<thead>
<tr>
<th>CHANNEL</th>
<th>PERCENT BUSY</th>
<th>CHANNEL</th>
<th>PERCENT BUSY</th>
<th>CHANNEL</th>
<th>BUSY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DASDs:**

1. Greater than 30% busy  
2. IOSQ time greater than 4 ms  
3. High activity greater than 15 per second

<table>
<thead>
<tr>
<th>VOLUME</th>
<th>REASON CODE</th>
<th>DEVICE ACTIVITY</th>
<th>AVE. RESP. RATE</th>
<th>AVE. IOSQ TIME</th>
<th>AVE. PEND TIME</th>
<th>AVE. DISC TIME</th>
<th>PERCENT DEVICE UTIL.</th>
<th>SHARED YES/NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR.</td>
<td>SERIAL</td>
<td>RATE</td>
<td>TIME</td>
<td>TIME</td>
<td>TIME</td>
<td>TIME</td>
<td>UTIL.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>Y/N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>Y/N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>Y/N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>Y/N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td>Y/N</td>
</tr>
</tbody>
</table>

*Figure 129. I/O Resource Worksheet*
C.3 Real Storage Resource Worksheet

Use the RMF paging reports, shown in Figures 14 to 16 on pages 52 to 54 for the calculation of real storage. The heading reference can be found under “Central Storage” in Figure 15 on page 53. Use these values for completing the worksheet (Figure 130 on page 382).
## REAL STORAGE WORKSHEET

### Storage Utilization:

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS Nucleus</td>
<td></td>
</tr>
<tr>
<td>LPA Frames Fixed</td>
<td></td>
</tr>
<tr>
<td>LSQA Frames</td>
<td></td>
</tr>
<tr>
<td>Total Fixed Frames</td>
<td></td>
</tr>
<tr>
<td>SQA Frames</td>
<td></td>
</tr>
<tr>
<td>Common Frames Fixed</td>
<td></td>
</tr>
<tr>
<td>Private Frames Fixed</td>
<td></td>
</tr>
<tr>
<td>Pageable Pool Frames</td>
<td></td>
</tr>
</tbody>
</table>

### User Defined Fixed Storage

<table>
<thead>
<tr>
<th>Storage</th>
<th>Size (K)</th>
<th>Storage</th>
<th>Size (K)</th>
<th>Storage</th>
<th>Size (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### System Paging:

#### A. Total System Paging Rate

- Total Demand Paging Rate
- Total Swap Paging Rate

#### B. Total Number of Swaps

- Number of TSO Swaps
- Number of Batch Swaps

#### C. Page Faults per Second

- IMS/ESA Control Region Page Fault Rate
- IMS/ESA DLS Region Page Fault Rate
- IMS/ESA DBRC Region Page Fault Rate
- All Dependent Regions Page Fault Rate

---

*Figure 130. Real Storage Resource Worksheet*
C.4 IMS Resource Worksheet

Use Figure 21 on page 71 and Figure 26 on page 83 with the output of the /DISPLAY POOL command to complete the analysis of the IMS resources.
### IMS System Transaction Profile:

<table>
<thead>
<tr>
<th>Item</th>
<th>TOTAL</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Trace Elapsed Time in Seconds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Number of Transactions Processed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Number of Transactions per Second</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Number of Schedules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E. Number of Transactions per Schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. Number of DL/I Calls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G. Number of DL/I Calls per Transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H. Number of Message Queue I/Os</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I. Number of MSGQ I/Os per Transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J. Number of OSAM I/Os</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Number of VSAM I/Os</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L. Number of Database I/Os per DL/I Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M. Number of Format I/Os</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N. Number of Format I/Os per Transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O. Number of ACBLIB I/Os</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P. Number of ACBLIB I/Os per Scheduling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q. Number of I/O per Transaction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Region Indicators:

| A. Mean Schedule Not IWAIT Time Greater than 12 ms.                   |       |     |
| B. Mean Schedule End to First DL/I Call Greater than 300 ms?          |       |     |

<table>
<thead>
<tr>
<th>TRANSACTION</th>
<th>TIME</th>
<th>TRANSACTION</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 131. IMS Resource Worksheet (1 of 2)*
IMS Region Occupancy:

Region 1 _______% Region 2 _______% Region 3 _______% Region 4 _______%
Region 5 _______% Region 6 _______% Region 7 _______% Region 8 _______%
Region 9 _______% Region 10 _______% Region 11 _______% Region 12 _______%

I/O IWAIT Times:

MEAN > 26 ms 3380  MAXIMUM > 300 ms
MEAN > 21 ms 3390

_________________________  ______________________
_________________________  ______________________

IMS System Transaction Profile:

<table>
<thead>
<tr>
<th>POOL</th>
<th>SIZE</th>
<th>HIGH USED PROBLEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. PSBP—PSB Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. DPSB—PSB Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. DMBP—DMB Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. MFP—Message Format Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E. QBUF—Message Queue Buffers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. DBAS—OSAM Database Buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G. DBAS—VSAM Database Buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H. SPAP—SPA Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I. PSBW—PSB Work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J. DBWP—Database Work Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. CIOP—BTAM Communications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L. HIOP—High Communications Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M. MAIN—Main Pool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 132. IMS Resource Worksheet (2 of 2)
C.5 IMS Application Program and Database Worksheet

The data for the application program and database worksheets is collected from the program summary report. An example can be found in Figure 22 on page 73.
## Most Active Application Programs

Trace Elapsed time _________ x _____ (No. of MPRs) = __________ (Region Elapsed Time)

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>ELAPSED</th>
<th>AVE. NUMBER</th>
<th>TRANS. SCHEDULES</th>
<th>DEQUEUED CALLS</th>
<th>I/O CALLS</th>
<th>IWAIT</th>
<th>CALLS/IWAIT</th>
<th>TRAN. I/O CALLS</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Most Active Databases:

Total Database I/O ____________

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>I/O OCCURRENCE</th>
<th>IWAIT MEAN</th>
<th>IWAIT TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 133. IMS Application Program and Database Worksheet
Appendix D. Special Notices

This publication is intended to help technical professionals understand how to use the IMS/ESA structure and how to tune the system. The information in this publication is not intended as the specification of any programming interfaces that are provided by IMS/ESA Version 5. See the PUBLICATIONS section of the IBM Programming Announcement for IMS/ESA Version 5 Release 1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these
names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ACF/VTAM
BookManager
CICS/ESA
Current
ES/3090
Hiperspace
ICING
IMS/ESA
MVS (logo)
MVS/ESA
OS/390
QMF
Resource Measurement Facility
RT
SP1
System/390
VTAM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.
Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 393.

- **IMS/ESA Version 5.1 Guide**, GG24-4302
- **IMS Fast Path Solutions Guide**, SG24-4301
- **IMS/ESA Data Sharing in a Parallel Sysplex**, SG24-4303
- **IMS/ESA Multiple Systems Coupling in a Parallel Sysplex**, SG24-4750
- **IMS/ESA Sysplex Data Sharing: An Implementation - Case Study**, SG24-4831
- **DB2 for OS/390 Terabyte Database: Design and Build Experience**, SG24-2072
- **DB2 for OS/390 Version 5 Performance Topics**, SG24-2213
- **DB2 for OS/390 Application Design Guidelines for High Performance**, SG24-2233
- **DB2 V4 Data Sharing Performance Topics**, SG24-4611
- **DB2PM Usage Guide**, SG24-2584
- **Locking in DB2 for MVS/ESA Environment**, SG24-4725

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. Order a subscription and receive updates 2-4 times a year at significant savings.

<table>
<thead>
<tr>
<th>CD-ROM Title</th>
<th>Subscription Number</th>
<th>Collection Kit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>System/390 Redbooks Collection</td>
<td>SBOF-7201</td>
<td>SK2T-2177</td>
</tr>
<tr>
<td>Networking and Systems Management Redbooks Collection</td>
<td>SBOF-7370</td>
<td>SK2T-6022</td>
</tr>
<tr>
<td>Transaction Processing and Data Management Redbook</td>
<td>SBOF-7240</td>
<td>SK2T-8038</td>
</tr>
<tr>
<td>AS/400 Redbooks Collection</td>
<td>SBOF-7270</td>
<td>SK2T-2849</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (HTML, BkMgr)</td>
<td>SBOF-7230</td>
<td>SK2T-8040</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (PostScript)</td>
<td>SBOF-7205</td>
<td>SK2T-8041</td>
</tr>
<tr>
<td>Application Development Redbooks Collection</td>
<td>SBOF-7290</td>
<td>SK2T-8037</td>
</tr>
<tr>
<td>Personal Systems Redbooks Collection</td>
<td>SBOF-7250</td>
<td>SK2T-8042</td>
</tr>
</tbody>
</table>

E.3 Other Publications

These publications are also relevant as further information sources:

- **IMS/ESA V5 Administration Guide: Database**, SC26-8012
- **IMS/ESA V5 Administration Guide: System**, SC26-8013
- **IMS/ESA V5 Administration Guide: Transaction Manager**, SC26-8014
- **IMS/ESA V5 Customization Guide**, SC26-8020
- IMS/ESA V5 Installation Guide; Volume 1, SC26-8023
- IMS/ESA V5 Installation Guide; Volume 2, SC26-8024
- IMS/ESA V5 Utilities Reference: Transaction Manager, SC26-8022
- IMS/ESA V5 Utilities Reference: Database, SC26-8034
- IMS/ESA V5 Utilities Reference: System, SC26-8035
- IMS/ESA V5 Diagnosis Guide and Reference, LY27-9620
- IMSASAP Monitor Summary and Analysis Program II, SB21-1793
- OS/390 MVS Workload Management Services, GC28-1773
- OS/390 Planning: Workload Management, GC28-1761
- OS/390 System Management Facilities, GC28-1783
- CICS/ESA V4.1 CICS—IMS Database Control Guide, SC33-1184
- CICS/ESA V4.1 CICS-Supplied Transactions, SC33-1168
- CICS/ESA V4.1 System Definition Guide, SC33-1164
- DATABASE2 V4 Administration Guide, SC26-3265
- DB2 PM Online Monitor User’s Guide, SH12-6165
- A User’s Guide to the IMS Monitor, GG66-3134
How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**
  To get LIST3820s of redbooks, type one of the following commands:
  
  TOOLS SENDTO EHONE4 TOOLSS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  
  To get BookManager BOOKs of redbooks, type the following command:
  
  TOOLCAT REDBOOKS
  
  To get lists of redbooks, type one of the following commands:
  
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  
  To register for information on workshops, residencies, and redbooks, type the following command:
  
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  
  For a list of product area specialists in the ITSO: type the following command:
  
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE

- **Redbooks Web Site on the World Wide Web**
  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**
  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**

- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

- **Internet Listserver**
  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

© Copyright IBM Corp. 1997
How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

<table>
<thead>
<tr>
<th>Country</th>
<th>IBMMAIL</th>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>usib6flpl at ibmmail</td>
<td><a href="mailto:usib6flpl@ibmmail.com">usib6flpl@ibmmail.com</a></td>
</tr>
<tr>
<td>Canada</td>
<td>calibmbkz at ibmmail</td>
<td><a href="mailto:lmannix@vnet.ibm.com">lmannix@vnet.ibm.com</a></td>
</tr>
<tr>
<td>Outside North America</td>
<td>dkibmbsh at ibmmail</td>
<td><a href="mailto:bookshop@dk.ibm.com">bookshop@dk.ibm.com</a></td>
</tr>
</tbody>
</table>

- **Telephone orders**

<table>
<thead>
<tr>
<th>Region</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States (toll free)</td>
<td>1-800-879-2755</td>
</tr>
<tr>
<td>Canada (toll free)</td>
<td>1-800-IBM-4YOU</td>
</tr>
<tr>
<td>Outside North America</td>
<td>(long distance charges apply)</td>
</tr>
<tr>
<td>(+45) 4810-1320 - Danish</td>
<td>(+45) 4810-1020 - German</td>
</tr>
<tr>
<td>(+45) 4810-1420 - Dutch</td>
<td>(+45) 4810-1620 - Italian</td>
</tr>
<tr>
<td>(+45) 4810-1540 - English</td>
<td>(+45) 4810-1270 - Norwegian</td>
</tr>
<tr>
<td>(+45) 4810-1670 - Finnish</td>
<td>(+45) 4810-1120 - Spanish</td>
</tr>
<tr>
<td>(+45) 4810-1220 - French</td>
<td>(+45) 4810-1170 - Swedish</td>
</tr>
</tbody>
</table>

- **Mail Orders** — send orders to:

<table>
<thead>
<tr>
<th>IBM Publications Customer Support</th>
<th>IBM Publications</th>
<th>IBM Direct Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Publications Customer Support</td>
<td>144-4th Avenue, S.W.</td>
<td>Sorømosevej 21</td>
</tr>
<tr>
<td>P.O. Box 29570</td>
<td>Calgary, Alberta T2P 3N5</td>
<td>DK-3450 Allerød</td>
</tr>
<tr>
<td>Raleigh, NC 27626-0570</td>
<td>Canada</td>
<td>Denmark</td>
</tr>
</tbody>
</table>

- **Fax** — send orders to:

<table>
<thead>
<tr>
<th>Region</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States (toll free)</td>
<td>1-800-445-9269</td>
</tr>
<tr>
<td>Canada</td>
<td>1-403-267-4455</td>
</tr>
<tr>
<td>Outside North America</td>
<td>(+45) 48 14 2207 (long distance charge)</td>
</tr>
</tbody>
</table>

- **1-800-IBM-4FAX (United States) or (+1)001-408-256-5422 (Outside USA)** — ask for:

  - Index # 4421 Abstracts of new redbooks
  - Index # 4422 IBM redbooks
  - Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

<table>
<thead>
<tr>
<th>Redbooks Web Site</th>
<th>IBM Direct Publications Catalog</th>
</tr>
</thead>
</table>

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

  **Redpieces**

  For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.
**IBM Redbook Order Form**

Please send me the following:

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First name

Last name

Company

Address

City

Postal code

Country

Telephone number

Telefax number

VAT number

Invoice to customer number

Credit card number

Credit card expiration date

Card issued to

Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.
**Glossary**

**archive process.** The process of archiving the IMS/ESA OLDS to a system log data set (SLDS) on either tape or DASD.

**arrival rate.** The rate at which transactions arrive at the IMS/ESA system if they could be put on the input queue instantaneously.

**bottleneck.** A place in the system where contention for a resource is affecting performance.

**commit point.** The place where the application program processing completes a unit of work. IMS/ESA makes permanent the changes that the program has made to the databases and makes the changed data available to other application programs.

**connect time.** The time at which a device is connected to a channel path to search or transfer data between the device and main storage.

**device service time.** The elapsed time from a successful start I/O to a successful I/O completion. This time does not include any time spent on the MVS queue waiting to start.

**disconnect time.** The time at which an I/O request is disconnected from the channel path during an I/O operation. It represents the time the device is in use but not transferring data to the processor.

**DL/I elapsed time.** The time spent by IMS servicing DL/I calls; includes IWAIT time.

**DL/I or DLS address space.** The address space that contains the IMS/ESA DL/I code and buffers created when the LSO=S option is specified.

**Exclusive Control.** A facility of IMS that allows for enqueuing and dequeuing of resources. The Program Isolation feature uses Exclusive Control.

**IMSASAP.** A Program Offering that analyzes the IMS Monitor trace file, summarizes the trace data, and produces additional reports.

**IMSPARS.** A Program Offering product that analyzes the IMS/ESA log file.

**I/O service time.** The elapsed time from when an I/O to a device is started until the I/O is successfully completed. This includes any time spent on the MVS queue waiting to start.

**IOSQ time.** The time an I/O request must wait on an IOS queue before an SSCH instruction can be issued. This delay occurs when a previous request to the same device or subchannel is in progress.

**IWAIT.** An IMS/ESA internal wait for I/O, pool space, or PI.

**latch.** A software switch in IMS similar to an MVS lock.

**MLPA.** Modifiable Link Pack Area. A part of MVS storage.

**monitoring.** Running a hardware or software tool to measure the performance characteristics of a system.

**MPR.** An IMS/ESA online message processing region.

**Not-IWAIT time.** Elapsed time minus IWAIT time for an event. Approximates CPU time if there is no interference.

**OLDS.** The IMS/ESA online log data set that contains the log records of all IMS/ESA activity.

**PEND time.** The time an I/O request must wait in the channel subsystem after the SSCH is accepted and before the first CCW is accepted by the DASD subsystem. This delay occurs when a previous request to the same device is in progress.

**preload.** A feature of IMS/ESA that loads modules into main storage at region initialization time so that they are available for execution without incurring any physical I/O by program fetch to bring the programs into real storage.

**profile.** A description of work in terms of DL/I calls, CPU, and I/Os for a transaction, application, dependent region, or system; that is, a transaction profile. It can also include elapsed times for events such as program elapsed time.

**program elapsed time.** The elapsed time after the first DL/I DC call in an application program to program termination.

**PST.** Partition Specification Table—one control block representing each message and batch message processing region in the IMS/ESA system and certain system tasks.

**region occupancy.** The percentage of total trace elapsed time that a message region is occupied. Region Occupancy includes Schedule/Termination, Schedule End to First DL/I call, and Program Elapsed times.
RMF. The MVS Resource Management Facility program product that is used to collect performance data on the MVS system.

RSR. IMS Remote Site Recovery, which provides remote recovery for IMS DB full function databases, IMS DB Fast Path databases, IMS TM Message Queues, and the IMS TM telecommunications network.

schedule approval. The point in time when the IMS control region has all resources available to schedule an application program in a message region and notifies the dependent region.

scheduling and termination. The elapsed time spent in the IMS control region scheduling and terminating an application program. Significant I/O activity is the loading of PSB and intent lists if they are not in main storage.

Schedule End to First DL/I Call. The elapsed time from schedule approval until the first DL/I call is issued by the application program. Significant I/O activity is the loading of the application program.

simple IMS/ESA checkpoint. A term used to denote when the IMS/ESA system control blocks are written to the system log.

SLDS. The system log data set, which is normally the tape data set that contains the archived OLDS. This data set is compatible with the IMS/ESA Version 1 Release 2 system log.

storage isolation. An MVS feature that protects real storage frames in either a private address space or in common from being stolen by the SRM.

sync point. A term used in IMS/ESA to denote the time when any committed records not previously written to the database are written to the WADS and database.

time-in-system. The time an IMS transaction is in the system from receipt of the input message to dequeue of the output message.

transaction. A message from a terminal or an application program that causes the application program logic to be executed.

unit of work. A distinct unit of processing that can be released by the application program for use by other programs in the system.

Virtual Fetch. A feature of MVS that IMS/ESA invokes to load an application program using the paging supervisor.

WADS. IMS/ESA write ahead data set, which holds the committed log records before writing them to the OLDS.
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACB</td>
<td>Access control block in VSAM. Application control block in IMS.</td>
</tr>
<tr>
<td>ACF/VTAM</td>
<td>Advanced communication function/virtual telecommunications access method.</td>
</tr>
<tr>
<td>ADS</td>
<td>(Read or write) ahead data set.</td>
</tr>
<tr>
<td>AIB</td>
<td>Application interface block.</td>
</tr>
<tr>
<td>AMS</td>
<td>Access method services.</td>
</tr>
<tr>
<td>APA</td>
<td>All points addressable.</td>
</tr>
<tr>
<td>API</td>
<td>Application programming interface.</td>
</tr>
<tr>
<td>APPC</td>
<td>Advanced program-to-program communication; also advanced peer-to-peer communication.</td>
</tr>
<tr>
<td>APPLID</td>
<td>VTAM application ID.</td>
</tr>
<tr>
<td>ARC</td>
<td>Automatic archiving count.</td>
</tr>
<tr>
<td>AWE</td>
<td>Asynchronous work element.</td>
</tr>
<tr>
<td>BALG</td>
<td>Balancing group.</td>
</tr>
<tr>
<td>BGWRT</td>
<td>Background write (IMS VSAM feature).</td>
</tr>
<tr>
<td>BLDS</td>
<td>Macro to build directories in partitioned data set memory.</td>
</tr>
<tr>
<td>blksize</td>
<td>Block size.</td>
</tr>
<tr>
<td>BMP</td>
<td>Batch message processing (IMS region).</td>
</tr>
<tr>
<td>BQEL</td>
<td>Buffer queue element.</td>
</tr>
<tr>
<td>BSAM</td>
<td>Basic sequential access method.</td>
</tr>
<tr>
<td>BTAM</td>
<td>Basic telecommunications access method.</td>
</tr>
<tr>
<td>CA</td>
<td>Control area.</td>
</tr>
<tr>
<td>CCB</td>
<td>Channel control block.</td>
</tr>
<tr>
<td>CCW</td>
<td>Channel command word.</td>
</tr>
<tr>
<td>CDE</td>
<td>Contents directory entry.</td>
</tr>
<tr>
<td>CI</td>
<td>Control interval.</td>
</tr>
<tr>
<td>CIDF</td>
<td>Control interval definition field.</td>
</tr>
<tr>
<td>CESS</td>
<td>Communication external subsystems pool (for DB2 connection).</td>
</tr>
<tr>
<td>CICS</td>
<td>Customer information control system.</td>
</tr>
<tr>
<td>CIOP</td>
<td>Communication input output pool (BTAM).</td>
</tr>
<tr>
<td>CLPA</td>
<td>Common link pack area (MVS).</td>
</tr>
<tr>
<td>CNT</td>
<td>Communication name block.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit.</td>
</tr>
<tr>
<td>CS</td>
<td>Central storage.</td>
</tr>
<tr>
<td>CSA</td>
<td>Common storage area.</td>
</tr>
<tr>
<td>CWAP</td>
<td>Communication work area pool.</td>
</tr>
<tr>
<td>DASD</td>
<td>Direct access storage device.</td>
</tr>
<tr>
<td>DBAS</td>
<td>Database buffer access statistics.</td>
</tr>
<tr>
<td>DBD</td>
<td>Database definition.</td>
</tr>
<tr>
<td>DBDGEN</td>
<td>Database definition generation.</td>
</tr>
<tr>
<td>DBFX</td>
<td>Number of Fast Path database buffers that are page fixed at IMS startup.</td>
</tr>
<tr>
<td>DBLDL</td>
<td>IMS procedure parameter, the maximum number of BLDDL entries.</td>
</tr>
<tr>
<td>DBT</td>
<td>Database tool.</td>
</tr>
<tr>
<td>DBRC</td>
<td>IMS/VS Database recovery control feature that is required for DASD logging and is recommended for managing database recovery and protecting databases from operational error.</td>
</tr>
<tr>
<td>DBWP</td>
<td>Database work area pool.</td>
</tr>
<tr>
<td>DC</td>
<td>Database communication.</td>
</tr>
<tr>
<td>DCB</td>
<td>Data set control block.</td>
</tr>
<tr>
<td>DCME</td>
<td>Dynamic cache management enhanced.</td>
</tr>
<tr>
<td>DD</td>
<td>Data set definition.</td>
</tr>
<tr>
<td>DDRN</td>
<td>DCB relative record number.</td>
</tr>
<tr>
<td>DEED</td>
<td>Data entry database.</td>
</tr>
<tr>
<td>DFLD</td>
<td>Device field.</td>
</tr>
<tr>
<td>DFSnnnnz</td>
<td>IMS message where nnnn are four numerics and the suffix z indicates the user action I = information, A = action, W = warning.</td>
</tr>
<tr>
<td>DFSxxxxx</td>
<td>IMS module, control block, table, or PROCLIB member.</td>
</tr>
</tbody>
</table>
DFW  DASD fast write.  
DL/I  Database language I.
DLISAS  DL/I separate address space.
DLS  DL/I separate address space.
DMB  Data management block (what the DBD becomes in the IMS control region).
DOVF  Dependent overflow.
DPSB  DL/I address space PSB.
DRA  Database resource adapter.
DRRN  Database relative record number.
DSA  Dynamics storage area.
DSME  Data space mapping entry.
DSML  Data space mapping list.
ECB  Event control block.
ECSA  Extended CSA (common storage area).
EMH  Expedited message handler.
EMHB  Expedited message handler buffer (terminal buffer).
EDPM  Enterprise performance data manager.
EPST  Extended partition specification table.
ESA  Enterprise systems architecture.
ESDS  Entry sequenced data set.
ESCON  Enterprise systems connection (architecture, IBM System/390).
ESF  Expanded storage (page) frames.
ESO  Expanded storage only.
ETO  Extended terminal option.
FAQE  Free/Allocated queue element.
FBFF  Free block frequency factor.
FIFO  First in, first out.
FLD  Field.
FSE  Free space element.
FSPF  Free space percentage factor.
FRE  Fetch request element.
GN  Get next.
GSAM  Generalized sequential access method.
GTF  Generalized trace facility.
GU  Get unique.
HD  Hierarchic direct.
HDAM  Hierarchic direct access method.
HDC  Hardware data compression.
HIDAM  Hierarchic indexed direct access method.
HIOP  High communication input output pool (VTAM).
HISAM  Hierarchic indexed sequential access method.
HSSP  High-speed sequential processing.
HSSR  High-speed sequential retrieval.
IBM  International Business Machines Corporation.
IFP  IMS Fast Path (IMS region).
IMS  Information management system.
IMSASAP  IMS/ESA monitor summary and system analysis program.
IMSPARS  IMS/ESA performance analysis and reporting system.
I/O PCB  Input/output program communication block, also called TPPCB.
IOS  Input output subsystem.
IOSQ  Queue wait time.
IOVF  Independent overflow.
IPCS  Interactive problem control system.
IPL  Initial program load.
IPS  Installation performance specification.
IRC  Interregion communication.
IRLM  IMS resource lock manager.
IRU  IMS resource usage.
ISC  Intersystem communication.
ISRT  Insert.
ITASK  IMS task.
ITSO  International Technical Support Organization.
JCL  Job control language.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JES</td>
<td>Job entry system (component of operating system JES2 and JES3).</td>
</tr>
<tr>
<td>KSDS</td>
<td>Key sequenced data set.</td>
</tr>
<tr>
<td>LGMGS</td>
<td>Long message queue data set; component of message queues.</td>
</tr>
<tr>
<td>LLA</td>
<td>Library lookaside.</td>
</tr>
<tr>
<td>LLE</td>
<td>Load list element.</td>
</tr>
<tr>
<td>LPA</td>
<td>Link pack area (MVS).</td>
</tr>
<tr>
<td>LRECL</td>
<td>Logical record length.</td>
</tr>
<tr>
<td>LRU</td>
<td>Least recently used.</td>
</tr>
<tr>
<td>LSO</td>
<td>Local storage option.</td>
</tr>
<tr>
<td>LSR</td>
<td>Local share resource.</td>
</tr>
<tr>
<td>LTERM</td>
<td>Logical terminal.</td>
</tr>
<tr>
<td>LWA</td>
<td>Log write ahead.</td>
</tr>
<tr>
<td>MADS</td>
<td>Multiple area data sets.</td>
</tr>
<tr>
<td>MDB</td>
<td>Most desirable block.</td>
</tr>
<tr>
<td>MFBP</td>
<td>Message format buffer pool.</td>
</tr>
<tr>
<td>MFS</td>
<td>Message format service.</td>
</tr>
<tr>
<td>MLPA</td>
<td>Modified link pack area (MVS).</td>
</tr>
<tr>
<td>MPP</td>
<td>Message processing program.</td>
</tr>
<tr>
<td>MPL</td>
<td>Multiprogramming level.</td>
</tr>
<tr>
<td>MPR</td>
<td>Message processing region (region in which MPPs are run).</td>
</tr>
<tr>
<td>MSC</td>
<td>Multiple systems coupling.</td>
</tr>
<tr>
<td>MSDB</td>
<td>Main storage database.</td>
</tr>
<tr>
<td>MTO</td>
<td>Master terminal operator.</td>
</tr>
<tr>
<td>MVS</td>
<td>Multiple virtual system.</td>
</tr>
<tr>
<td>NBA</td>
<td>Normal buffer allocation (DEDBs).</td>
</tr>
<tr>
<td>NCP</td>
<td>Network control program.</td>
</tr>
<tr>
<td>NSR</td>
<td>Nonshared resources.</td>
</tr>
<tr>
<td>NVS</td>
<td>Nonvolatile storage.</td>
</tr>
<tr>
<td>OBA</td>
<td>Overflow buffer allocation (DEDBs).</td>
</tr>
<tr>
<td>OLDS</td>
<td>Online log data set.</td>
</tr>
<tr>
<td>OSAM</td>
<td>Overflow sequential access method.</td>
</tr>
<tr>
<td>PARLIM</td>
<td>Parallel limit.</td>
</tr>
<tr>
<td>PARMLIB</td>
<td>MVS parameter library data set.</td>
</tr>
<tr>
<td>PCB</td>
<td>Program communication block; can be TPPCB (I/O PCB) or DBPCB (database).</td>
</tr>
<tr>
<td>PCF</td>
<td>Physical child first.</td>
</tr>
<tr>
<td>PCL</td>
<td>Physical child last.</td>
</tr>
<tr>
<td>PDS</td>
<td>Partition data set.</td>
</tr>
<tr>
<td>PI</td>
<td>Program isolation.</td>
</tr>
<tr>
<td>PLC</td>
<td>Process limit count.</td>
</tr>
<tr>
<td>PLPA</td>
<td>Pageable link pack area (MVS).</td>
</tr>
<tr>
<td>PROCLIB</td>
<td>Procedure library data set; IMS procedures are in IMSESA.PROCLIB and MVS procedures are in SYS1.PROCLIB.</td>
</tr>
<tr>
<td>PROFS</td>
<td>Professional Office System.</td>
</tr>
<tr>
<td>PSB</td>
<td>Program specification block, made up of PCBs, both I/O and database.</td>
</tr>
<tr>
<td>PSBP</td>
<td>PSB pool.</td>
</tr>
<tr>
<td>PSBW</td>
<td>PSB work pools.</td>
</tr>
<tr>
<td>PSSR</td>
<td>Physical sequential reload.</td>
</tr>
<tr>
<td>PST</td>
<td>Partition specification table.</td>
</tr>
<tr>
<td>PTF</td>
<td>Physical twin forward, also program temporary fix.</td>
</tr>
<tr>
<td>QBLKS</td>
<td>Queue blocks data set, component of message queues.</td>
</tr>
<tr>
<td>QMGR</td>
<td>IMS queue manager.</td>
</tr>
<tr>
<td>RACF</td>
<td>Resource access control facility.</td>
</tr>
<tr>
<td>RAID5</td>
<td>Redundant array of independent disks format five (5).</td>
</tr>
<tr>
<td>RAP</td>
<td>Root anchor point.</td>
</tr>
<tr>
<td>RBA</td>
<td>Relative block address.</td>
</tr>
<tr>
<td>RDF</td>
<td>Record definition field.</td>
</tr>
<tr>
<td>RES</td>
<td>Resident.</td>
</tr>
<tr>
<td>RLDS</td>
<td>Recovery log data set.</td>
</tr>
<tr>
<td>RMF</td>
<td>Resource management facility.</td>
</tr>
<tr>
<td>RMFCS</td>
<td>RMF client/server.</td>
</tr>
<tr>
<td>RNL</td>
<td>Resource name list.</td>
</tr>
<tr>
<td>RPL</td>
<td>Request parameter list.</td>
</tr>
<tr>
<td>RTR</td>
<td>Response transaction.</td>
</tr>
<tr>
<td>SB</td>
<td>Sequential buffering.</td>
</tr>
<tr>
<td>SCD</td>
<td>System contents directory.</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>SIO</strong></td>
<td>Sequential I/O.</td>
</tr>
<tr>
<td><strong>SDEP</strong></td>
<td>Sequential dependents.</td>
</tr>
<tr>
<td><strong>SHMGS</strong></td>
<td>Short message queue data set, component of message queues.</td>
</tr>
<tr>
<td><strong>SLDS</strong></td>
<td>System log data set.</td>
</tr>
<tr>
<td><strong>SMB</strong></td>
<td>Scheduler message block.</td>
</tr>
<tr>
<td><strong>SLR</strong></td>
<td>Service level reporter.</td>
</tr>
<tr>
<td><strong>SLUP</strong></td>
<td>Service logical unit parameter.</td>
</tr>
<tr>
<td><strong>SLUTYPEP</strong></td>
<td>Service logical type parameter.</td>
</tr>
<tr>
<td><strong>SMF</strong></td>
<td>System measurement facility.</td>
</tr>
<tr>
<td><strong>SNA</strong></td>
<td>System network architecture.</td>
</tr>
<tr>
<td><strong>SPA</strong></td>
<td>Scratch pad area.</td>
</tr>
<tr>
<td><strong>SPAP</strong></td>
<td>Scratch pad area pool.</td>
</tr>
<tr>
<td><strong>SPQB</strong></td>
<td>Scratch pad queue block.</td>
</tr>
<tr>
<td><strong>SPIE</strong></td>
<td>Specify program interruption exit.</td>
</tr>
<tr>
<td><strong>SQA</strong></td>
<td>System queue area.</td>
</tr>
<tr>
<td><strong>SRB</strong></td>
<td>Service request block.</td>
</tr>
<tr>
<td><strong>SRM</strong></td>
<td>System resources manager (MVS).</td>
</tr>
<tr>
<td><strong>SSA</strong></td>
<td>Segment search argument.</td>
</tr>
<tr>
<td><strong>SSCH</strong></td>
<td>Start subsystem channel.</td>
</tr>
<tr>
<td><strong>STAE</strong></td>
<td>Specify task asynchronous exit.</td>
</tr>
<tr>
<td><strong>SVC</strong></td>
<td>Supervisor call.</td>
</tr>
<tr>
<td><strong>TCB</strong></td>
<td>Task control block.</td>
</tr>
<tr>
<td><strong>TCP/IP</strong></td>
<td>Transmission control protocol/Internet protocol.</td>
</tr>
<tr>
<td><strong>TM</strong></td>
<td>Transaction manager.</td>
</tr>
<tr>
<td><strong>TPN</strong></td>
<td>Transaction process name.</td>
</tr>
<tr>
<td><strong>TPPCB</strong></td>
<td>Input/Output program communication block, also called I/O PCB.</td>
</tr>
<tr>
<td><strong>TSO</strong></td>
<td>Time sharing option.</td>
</tr>
<tr>
<td><strong>UOW</strong></td>
<td>Unit of work.</td>
</tr>
<tr>
<td><strong>VIO</strong></td>
<td>Virtual I/O (input/output).</td>
</tr>
<tr>
<td><strong>VLF</strong></td>
<td>Virtual lookaside facility.</td>
</tr>
<tr>
<td><strong>VSAM</strong></td>
<td>Virtual sequential access method.</td>
</tr>
<tr>
<td><strong>VSO</strong></td>
<td>Virtual storage option.</td>
</tr>
<tr>
<td><strong>VSCR</strong></td>
<td>Virtual storage constrain release.</td>
</tr>
<tr>
<td><strong>VTAM</strong></td>
<td>Virtual telecommunications access method.</td>
</tr>
<tr>
<td><strong>VTCB</strong></td>
<td>VTAM terminal control block.</td>
</tr>
<tr>
<td><strong>VTOC</strong></td>
<td>Volume table of content.</td>
</tr>
<tr>
<td><strong>WADS</strong></td>
<td>Write ahead data set.</td>
</tr>
<tr>
<td><strong>WFI</strong></td>
<td>Wait for input.</td>
</tr>
<tr>
<td><strong>WKAP</strong></td>
<td>Work area pool.</td>
</tr>
<tr>
<td><strong>WLM</strong></td>
<td>Work load manager.</td>
</tr>
<tr>
<td><strong>XRF</strong></td>
<td>IMS extended recovery facility.</td>
</tr>
</tbody>
</table>
Index

Special Characters
/ASSIGN (ASSIGN command) 212
/CHE (checkpoint command) 18
/DBR (database recover) 24, 28, 93, 189, 196
(DIS (display command) 89, 162, 186—196
  display active 29
  message format pool 132, 186
  message queue pool 94
  POOL ALL 19, 55, 56, 89, 90, 93, 354
  POOL DBAS 278
  POOL FPDB 277
  POOL MAIN 204
  PSBW 194
/ERE (emergency restart) 97
/STA (start command) 26, 28, 225

Numerics
0Cx abends 224, 225
477 abends 97
775 abends 199, 354
777 abends 66, 224, 288, 290
778 abends 66, 296, 297
801 abends 97
804 abends 97

A
abbreviations 399
ACBLIB 138, 161, 188—192
  BLDL 193
  BLKSIZE 189
  data set placement 134
  DMB 10, 189
  I/Os 78, 79, 189, 192, 369, 371
  load 10, 11
  PSB 10, 189, 192, 193
  tuning 188
  access method 130, 280
ACF/VTAM
  See VTAM
acronyms 399
alternate PCB
  See PCB
API
  explicit 298—301, 336
  implicit 298
APPC/IMS 176, 297
  APPC/MVS 299
EMH buffer 184
application resources
  BMP processing 291
  application resources (continued)
    design for performance 279
    program analysis 105
    program design 287—289
    program elapsed time 106, 115
    program preload 226
    program structure 289
ASRT call 103

B
base profile 1, 6
batch DL/I 5, 6, 49, 140, 142
  BLDL 142
    buffer usage 251
    nonswapable 141
    performance 244
    swapping 140, 141
BGWRT 243, 244
  DFSVSMxx 78, 244
  execution parameter 24
  Program Summary report 79
bibliography 391
BLDL 11, 222
  entries 142, 223
  initialization 6
  list 207, 223, 288, 297
block loader 193, 369
block mover 29, 120, 189
BMP 142, 177, 199, 241—347
  BGWRT 78
    checkpoint frequency 15, 78, 294
    checkpoint message suppression 24
    CHKPT 15
    database updates 244
    DBCTL 321, 325
    dispatching priority 119, 234, 294, 337
    elapsed time 142
    Fast Path 15
    message-driven 15
    never ending 158
    PI enq/deq pool 354
    planning 294
    processing 238, 239, 291, 294
    PST 26
    region occupancy 114
RESLIB 222
  Schedule End to First DL/I Call 81
storage isolation 159
SYNC 15, 291
BQEL 256

© Copyright IBM Corp. 1997
BTAM 14, 98, 99, 176
buffer pool purge 256
buffer pool storage 161
See also dynamic pool manager, database buffer pool
See also Fast Path buffer pool, OSAM buffer pool
See also VSAM buffer pool, OSAM buffer pool
Buffer Queue Element (BQEL) 256
bypass MFS 10

See also dynamic pool manager, database buffer pool
See also Fast Path buffer pool, OSAM buffer pool
See also VSAM buffer pool, OSAM buffer pool

C

capacity planning 7, 49, 64, 352
CESS pool 168, 205
checkpoint call 7, 18, 19, 64, 292
See also DL/I call
action taken 292
archive 28
BMP 15, 291—294
calls 74
data written 294
description 291
extended 291
frequency 15, 78, 291, 294
GSAM 292
IMSPARS 189
interval 28, 348
MSDB 245
OLDS 28
simple 32, 97, 178
symbolic and basic 291
checkpoints (system) 151, 153
CHKPT
See also checkpoint call
CHTS 27
CIOIP pool 89, 98, 162, 176
class scheduling 3, 114, 208, 234, 301, 347, 373
algorithm 64
preload 226
reevaluate 112
region occupancy 111
CLPA 225
COBOL
IGZERRE environment initialization 235
LIBKEEP 235
options 234
preload 225
Common Execution Environment 235
communication pools
See also CIOIP, HIOP, SPAP
Communications Line reports
Communications IWAIT 102
Communications Summary 99
Line Function 99
control region CPU usage 352

core region execution parameters
ARCHIVE 26
EXVR 23
Hash Tables 27
LSO 25
PRLD 23
PST 26
RES 26
SAV 26
SRCH 23
VAUT 25
VSAMPLS 24
VSPEC 24, 240

CPSLOG
IMSCTF macro 27
CPU resource 41, 66, 117, 119, 121, 122, 352
general data 40
CSA 38, 51—53, 60, 190, 199
CSAPSB 89
ENQ/DEQ 25
intent list 191
LSO 26, 190, 191
MSDB 245
pageable frames 38
paging 110, 141, 142
PSB Pool 153, 190
PSBP 93
RMF 60
storage isolation 143
storage requirement 234
CSAPSB 89, 190
CSAPSB pool tuning 190
CWAP pool 97, 162, 176

data collection 40, 61, 69, 105, 106
data management block
I/Os 10, 189, 369
IWAITs 93
loading 10
pool 12, 26, 93, 111, 120, 188, 189, 196
pool tuning 196
resident 10, 26, 196
data set groups 283, 285, 286
database buffer pools 239
database resource adapter
See also DRA
database resources 105
See also DEDB, HDAM, HIDAM, HISAM, MSDB
access method 280, 282
compaction 284
data set groups 283, 285, 286
database I/O analysis 106
design 1, 63, 129, 239, 279
logical relationships 286
database resources (continued)
  physical twin chains 283
  Program Isolation 66, 88, 112, 119, 121, 197—199,
   208, 284, 321, 354, 356
  reorganization 246, 283
  secondary indexing 24, 103, 154, 280, 283, 286, 287
  segmentation 280, 282
  structure 131, 239, 282
  variable length segments 284
  database work pool tuning 197

DATABASE2
  See DB2

DB2 325, 338
  BIND ACQUIRE parameters 334
  BIND ISOLATION parameters 335
  BIND RELEASE parameters 334
  deadlock with IMS 200
  locking 333
  LOCKSIZE parameter 333
  timeout 200
  type 2 indexes 334

DBCTL 120, 315
  APF 324
  BMP 321, 325
  DB2 325
  DEDB 324, 326
  dispatching priorities 327
  DMB 324
  FPBUF 326
  IMS Monitor 321
  PI 321
  PSB 322

DBFULTA0 5, 20

DBFULTA0 (Fast Path Log Analysis utility) 62
  LGNR 202

DBLDL
  See BLDL

DBPCB
  See PCB

DBRC 33, 118, 305, 310, 311
  dispatching priority 119

DBT HD Tuning Aid 61, 62, 63, 67, 68, 69, 246, 280, 283

DBWP pool 92, 97, 197
  deadlock 291
    events 290
    occurrences 290
    PSB 290
    shared DASD 310
    U777 abend 290
  DEDB 15, 203, 246, 280, 291, 324
    See also Fast Path
      areas 248, 280
      HDAM 280

DEDB (continued)
  I/O priority 133
  LGNR 202, 203
  subset pointers 283
  sync point phase one 15
  tools 62, 63
  dependent region 113, 134, 140, 223, 233
  DFSCNTL DD statement 25
  DFSDFRx 153
  DFSERA10 321
  DFSERA10 (Log Tape Formatting Print Program) 62, 64, 66
  DFSFiXxx 17, 18, 23, 55, 56, 143, 154, 192, 223
  DFSHDC40 281
  DFSILTA0 (Log Transaction Analysis utility) 62, 64
  DFSMPLxx 17
  DFSMRCL0 55, 142
  DFSPPBxxx 17, 89, 179, 190
  DFSPIRP0 321
  DFSSPMxxx 17
  DFSUTR20 (IMS Monitor Report Program) 18, 63, 361, 368, 373
  DFSVFLxx 17, 230
  DFSVSMxxx 24, 153
    BGWRT 24
    OSAMOP 24
    Overrides PST parameter 26
    SBONLINE 25
    STRINGMX 24
    VSAMFIX 24
    VSAMPS 24
  DFW 30, 31, 49, 202
  DIE 368
  disabled interrupt exit
    See DIE
  dispatching priority 8, 33, 37, 41, 42, 46, 111, 112,
    118—119, 120, 131—135, 228, 234, 294, 327, 337, 352, 371
    BMP 119, 234, 294, 337
    control region 111, 117, 118, 119, 337
    DB2 address spaces 337
    DBRC address space 118, 119, 337
    DLS address space 118, 119, 337
    GRS 118
    GTF 118, 119
    I/O IWAIT time 132
    IRLM address space 118, 119, 337
    JES 118
    MPR 119, 234, 337
    RMF 37, 118, 119
    VTAM 118, 119
    within IMS 111
  DL/I Call Summary 18, 64, 102, 104, 115, 283, 286, 287
    I/Os per call 103
    NOT-IWAIT time 102, 103
DL/I calls
  CHK P 15, 198, 292
  get next (GN) 198, 282, 288
  get unique (GU) 15, 66, 209, 210, 221, 225, 288, 289, 347
  insert (ISRT) 281, 288
  replace (REPL) 198
  restart (XRST) 291
  SYNC 15
DLIPSB 89, 190
DLIPSB pool tuning 190
DLS address space 55, 78, 81, 118, 119, 137, 138, 139, 243
DMB
  See data management block
DOPT PSB option 193, 211
DPRTY 118, 119
DRA (database resource adapter) 324
DSPURI01 310
dynamic allocation 28, 29, 63, 89, 189
  of IMS monitor 63
dynamic BLDL
  See BLDL
dynamic pool manager 89, 98, 162—173, 176

E
ECSA
  See CSA
EDPM 36
EMH
  APPC buffer allocation 184
  buffer pool 182
  dynamic pool management 162
  pool space shortage 173
ENQ/DEQ
  See PI ENQ/DEQ
EPDM 38
EPST page fixing 154
ERMRMFxx 18
ESCON 30, 49
  use with IMS 30
explicit API
  See API
express PCB
  See PCB
external conflict 213

F
Fast Path
  BALG 14, 15
  buffers 248
  DBBF 248
  DBFX 248
  DE DB 15, 62, 63, 203, 246, 280, 283, 291, 324
  Fast Path (continued)
  I/O priority 133
  MSDB 15, 245
  NBA 246—249
  OBA 246—249
  OTHREAD 16, 246, 248
  Fast Path database 245
  DE DB 15, 62, 63, 203, 245, 280, 283, 291, 324
  DE DB as first choice 280
  MSDB 15, 245
  Fast Path Log Analysis utility
    See DBFULTA0
  Fetch Request Elements
    See FRES
  FIX list 17, 18, 50, 55, 89, 142, 192
    IMS 55, 56
    MVS 142
  fixed frames 38, 51, 55, 142, 159
  free space 94, 127—129, 131, 132, 186, 196, 285
  FREEMAIN 230
  FRES 94, 132, 186, 188
    tuning 186

G
GETMAIN 173, 230
glossary 397
GSAM 292, 295
RSA warning 296
GTF 5, 36—38, 126
  DB2 337
    dispatching priority 118
    loading subroutines 225
  OSAM batch 25
    paging 93
  SMF 35
  VTAM 36

H
Hash Tables 27
HDAM 24, 67, 280—284
  DE DB 280
  DE DB as first choice 280
  randomizer 24, 67, 281
  RAPS 67, 280, 281, 282
  reentrant 282
  reusable 282
  sequential randomizer 281
HIDAM 67, 103, 282—284
HIOP pool 30, 89, 98, 162, 176
Hiperspace 24, 241—244
  benefits 241—243
  buffers 241—244
  using 241, 244
IMSPARS reports
  CPU Usage 66, 352
  DASD log statistics 354
  Database Activity 65
  DMB pool statistics 354
  internal resource usage 354
  PI Enq/Deq 354
  Queue Manager 66
  region histogram 65, 348
  Resource Usage 65
  transit log 347
  Transit Time Analysis 65, 347

In-Q Time
  intent conflict 213
  intent failure 216
  intent list 10, 25, 26, 78, 155, 189—193, 369
  internal conflict 213

IOPCB
  See PCB

IOQ parameter 133

IRLM 154, 199—344
  address space 118, 137, 159
  block level sharing 118
  DBCTL 315
  dispatching priority 119
  log records 66
  PI 118, 199

ISSUE681 24

VSAMCLS 24

ITASK 99, 132, 154

IWAIT
  ACBLIB NOT-IWAIT 111
  DL/I NOT-IWAIT 283
  I/O 10, 78, 88, 103, 110, 112, 114, 123, 130
  NOT-IWAIT 10, 63, 81, 102, 103, 111
  region 46, 78, 80, 88, 93—97, 106, 177, 321
  scheduling NOT-IWAIT 111, 118

J

JCL 243
  BMP 222
  classes 133
  control region 17
  dependent region 17, 231
  DFSPBxxx 190
  IMS Monitor 64
  LSO 190
  sequential buffering 239
  JES 29, 118, 292
  dispatching priority 118
  JOBLIB 23, 222—225
Latch statistics 66, 120
LE/370 235
LGMSG 65, 102, 134, 177—182, 289, 368
LHTS 27
LIBKEEP COBOL runtime option 235
link pack area
See CLPA, LPA, MLPA, PLPA
LOCKMAX 66, 284
log analysis
See DBFULTA0, DFSILTA0
log buffer 202
log I/Os 15, 25, 31
log reduction 201
Log Transaction Analysis utility
See DFSILTA0
Log Write-Ahead (LWA) 257
long physical twin chain 283, 286
LPA 18, 38, 51, 55, 57, 60, 223
fixed frames 38, 51
pageable frames 38
LSO option
LSO=S 25, 93, 143, 155, 190, 191
LSO=Y 25, 26, 93, 154, 159, 190
VSAMPL 154
LU 6.2 176
LUMC pool 176
LUMP pool 176
LWA 257

MACRO
BLDL 193
CPLOG keyword 27
DFSPRP 324
GETMAIN 230
IMSCFT 27
PGRLSE 230
SYSEVENT 140
TERMINAL 29
TRANSACT 210, 212
Master Terminal Operator
See MTO
MAXRGN 212
Message format pool tuning 184
FREs 186
Message format services 10, 95, 288
message prefix 179
message priming 74, 103, 288
Message Processing Program 123, 330, 347
checkpoint 291
Fast Path 15
long running programs 244
MAXRGN 212

Message Processing Program (continued)
MODE=SNGL 211
PGMLIB 222
sequential buffering 239
sync points 198
Message Processing Region 3, 232, 331
BLDL 223
dispatching priority 119
LLA 232
number of 113
paging 110
PWFI 210
message queue 15, 89, 94, 102, 113, 177—182, 368, 371
buffers 10
GU 66, 209, 210, 221, 225, 289, 347
I/Os 10, 15, 74, 79, 111, 177, 182, 288, 368, 371
LGMSG 65, 102, 134, 177—182, 289, 368
multiple data sets 134
multiple message queue 178
page fix 23
pool statistics 74
primed message 70, 103
QBLKS 134, 181, 368
QBUF 11, 89, 179
QC status 209, 210
SHMSG 65, 102, 134, 177—182, 368
single segment message 288
utilization 19, 21, 65, 347, 368
wait 208
message queue pool
general tuning 177
Message Queue pool tuning 177
buffer LRECL sizes 181
SPA size 182
MFS
See Message format services
MLPA 23, 159, 225
MODE=MULT 289
MPP
See Message Processing Program
MPR
See Message Processing Region
MSC message prefixes 180
MSDB 15, 245
See also Fast Path
MTO 63, 209, 291, 292, 323

NBA 246—249, 326
buffers used 249
EXEC parameter 248
shortage 249
NCP 18
NHTS 27
Non-volatile storage 202
nonrecoverable database 201

O
OBA 246—249
buffers used 249
EXEC parameter 248
latch 249
shortage 249
OLDS 16, 31—34, 105, 121, 201
allocation 32
archive 28, 31, 33
checkpoint 28
data set placement 134
DFW 31, 202
dual 32
example 244
performance 28, 31, 202
physical block 32
WADS 33
XRF 34
OSAM buffer pool 24, 71, 89, 94, 154, 354
OSAMOP 24

P
page fixing 8, 137
CIOP 153, 170, 176
control blocks 138, 144, 154
dependent region 233
DFSFIXxx 23, 143
DL/I 143
EMHB 153, 170, 176
enqueue pool 199
IMS control blocks 143, 153
IMS nucleus 143
IMS pools 153
message queue buffer pool 23
NBA 248, 249
OSAM Buffers 24
pools 137, 138, 153, 192, 193
PSB pool 193
resident DMB 196
SPAP 153, 170, 176
storage isolation 56, 143
versus storage isolation 138
VSAM Buffers 24
VTCBs in DREF storage 153
paging
data sets 141
dependent region 233
page faults 60, 138
swapping 57, 140
system 56, 139
parallel scheduling 12, 209, 212, 289
objectives 212, 289
PSB 209
SCHDTYPE=PARALLEL 155
WFI 209
PARDLI 236
PARLIM 212
/ASSIGN 212
keyword 212
parmlib
See SYS1.PARMLIB
PCB
alternate PCB 190
DBPC 239
DBPCB 25, 155, 190, 191, 239, 284, 292, 296
express PCB 289, 296
IOPCB 93, 190, 209, 221, 225, 289
performance
data collection 40, 61, 69
I/O IWAIT TIMES 10, 69, 78, 88, 110, 112, 130,
132, 361, 368, 373
idle for intent 88
information sources 17
Key Performance Indicators 4, 109, 116
LGNR 203
Mean I/O IWAIT Times 112
number of ACBLIB I/Os 78, 79, 189, 192, 369, 371
number of BMPs 294
number of buffers 248
number of database buffers 94, 97, 241
number of Database I/Os 78, 239, 368
number of database updates 65
number of database updates. 292
number of DL/I calls 2, 63, 74, 105, 106, 115, 288,
362
number of Format I/Os 78, 79, 368
number of FRES 94, 132, 186, 188
number of Hiperspace buffers 243
number of I/Os 2, 9, 10, 78, 79, 106, 107, 112,
123, 134, 189, 192, 283, 322, 368, 369, 371
number of IWAITs 63, 107, 128
number of LGNR 202, 203
number of Message Queue buffers 94, 179
number of Message Queue I/Os 10, 15, 74, 79,
368
number of MVS resources 51
number of OLDS 26, 32, 354
number of OLDS buffers 32
number of OSAM I/Os 74, 78, 79, 368
number of PGMLIB I/Os 79, 134, 371
number of PSTs 4, 26
number of RECANY 66, 98
number of regions 112—115, 121, 140, 194, 212,
242, 347
number of schedules 74, 115
number of threads 322, 324
performance (continued)
number of transactions 63, 69, 105, 361, 362
number of VSAM I/Os 76, 78, 79, 368
number of waits 149
objectives 1—3
Region Occupancy 82, 111
Schedule End to First DL/I Call 81, 114, 123, 221
Schedule NOT-IWAIT Time 81, 371
Schedule/Termination NOT-IWAIT Time 111
tools IMS 61
tools MVS 35
tuning data required 35, 61
performance objectives 1—3
establish 1
PGMLIB 79, 134, 222, 223, 371
PI 12, 63, 66, 88, 113, 118, 197—199, 291, 321
contention 116
control 199
ENQ/DEQ storage 154
enqueue 113, 119, 197, 199, 208, 354, 356
IRLM 118, 199
IWAIT 88, 284
locking 118
PI Trace 63
trace 66, 88, 321
PI ENQ/DEQ (Program Isolation) 88, 197
contention 88
control blocks 199, 292
IRLM 118, 199
IWAIT 88, 284
storage 25, 88, 154, 292
trace 63, 88, 321
wait time 12
PI Trace 63, 66, 88, 321
PIMAX 292
PLPA 18, 38, 55, 141, 155, 159, 223—225
pool name matrix 89
pool space analysis 89
CIOP 176
DBWP 197
DMB 188, 196
DPSB 188, 190
HIOP 176
Message Format Pool 94
MFS 184
MSGQ 177
PSB 188
PSBP 190
PSBW 188, 194
space calculation 161
SPAP 97, 176
VSAM 240
WKAP 97
pool space failure 213
preload 24, 192, 221, 223, 230—232, 233
candidates 81
preloading (continued)
class scheduling 226
control region 17
dependent region 17
HDAM randomizer 24
preload list 297
U777 abend 290
U778 abend 297
PREOPEN (DBRC) 307
priority conflict 213
problem
analysis 344
avoidance 4
capacity 1
communication pools 30
database allocation 29
database buffers 251
dispatching 131
DL/I 140
identification 42, 46, 64, 78, 81, 88, 116, 123, 126, 129
input 30
isolation 332, 344, 373
logging 32, 202, 279
paging 141
performance 1, 5, 35, 44, 61, 109, 115, 116, 245, 246
placement 130
real storage 141
scheduling 111, 289
transaction 64
PROCOPT 199
PROCOPT=GOT warning 295
Program Isolation
See PI
Program load libraries 79, 134, 222, 223, 371
program specification block 97, 105, 106, 118, 188, 189, 190, 199, 239, 290, 322
buffer stealing 192
cloning 192
DOPT 193, 211
I/Os 10, 78, 93, 189—194, 369
intent list 10, 25, 26, 78, 155, 189—193, 369
IWAIT TIME 106
name 67
pool 25, 26, 93, 111, 153, 155, 188—194, 294, 324, 354
pool space management 192
pool tuning 190
PSBW pool 97, 188, 194
resident 26, 192, 193
scheduling 192
trace 63
PSB
See program specification block
PSB work pool 97
PSBW pool 97
PSBW pool tuning 194
Pseudo-Wait-for-input
  See PWFI, WFI
PST parameter 26
purging database buffer pools 256
PWFI 210

Q
QBLKS 134, 181, 368
QBUF 11, 89, 179
QMF 336
QMGR 74, 121, 122
QNAME 310
QSAM 11
quick reschedule 211

R
RAID5 DASD 31, 34
RAPS 67, 280, 281, 282
read-only threads (DB2) 330
real storage resource 50
  IMS storage always fixed 55
  MVS user fixed storage 55
Real/Virtual Storage
  overview 137
RECANY (receive any) 10, 30, 66, 98
  buffer 10, 98, 176
  statistics 66
reentrant 225, 231, 281, 282
  HDAM 282
  U778 abend 297
region idle 214
region IWAIT 46, 78, 80, 88, 93—97, 106, 177, 321
region occupancy 82, 88, 111—115
resident DMB 10, 26, 196, 294
resident intent list 10, 25, 155, 189—193
resident PSB 26, 155, 189—193
RESLIB 64, 142, 222, 324
resource cleanup
  See DFSMRCRL0
response time (continued)
  I/O 32, 44, 46, 137, 179, 188, 226
  JES3 29
  MAXRGN 212
  Message Format Service 132
  objectives 3
  page faults 143
  page fixing 138, 143, 144
  paging 50, 139, 140, 159, 179, 233
  parallel scheduled 212
  PARLIM 212
  pools 89
  region occupancy 113
  resources 3
  ROLL call 297
  storage 137
  system 106
  VLF/LLA 226, 227
  WADS 121
  WFI 111
reusable 231—232, 282
  HDAM 282
  RMF 5, 18—20, 36—37, 41—55, 56, 57, 60, 246
  ROLL/ROLB call 296
    differences 296
    use of ROLB call 297
    use of ROLL call 297
root anchor points
  See RAPS
  RSA (GSAM) 296

S
SAPs 149
SAV parameter 26
  CHTS 27
  LHTS 27
  NHTS 27
  UHTS 27
  SAV= 149
Save Area Prefix
  See SAPs
SBONLINE 25
SCHD 213
latches 214
  responsibilities 214
Schedule and Termination 29, 78—81, 82, 111, 139,
  373
Schedule End to First DL/I Call 12, 63, 81, 82, 114,
  115, 123, 135, 207—223, 371
Schedule NOT-IWAIT time 81, 111, 118, 371
Scheduling failures 213
SDEP 248, 280
sequential dependents
  See SDEP
SHMSG 65, 102, 134, 177—182, 368
SLDS 33
SLR 36, 38
SMB 10, 90, 121, 181
SMF 35, 37, 209, 229, 323, 338
DB2 337
GTF 35
SPA size 181, 182, 289
space failure 213
SPAP pool 89, 97, 162, 176
status codes
QC status 209, 210
STEPLIB 23, 222—225, 231
storage isolation
common area 159
CSA 143
defining 158
dependent region use 159
description 158
DL/I address space 143
IMS nucleus 143
monitoring the use 160
paging rates 159
reports required 158
storage resource
paging 139
storage isolation 143
STRINGMX 24
Supervisor Call
See SVC
SVC 28, 187, 222
swapping 57
SYS1.LINKLIB 55, 222
SYS1.PARMLIB 18, 55, 129, 142, 222, 227—228, 243
COFVLFxx 18, 227, 228
COMMNDxx 18
CSVLLAx 18, 227
IEAFIXxx 18, 55, 142, 223
IEAICSxx 18
IEAIPSxx 18, 243
IEALODxx 223
IEALPxxx 18, 223
IEAOPTxx 18
IEASYSxx 18, 141
IECIOSxx 18
LNKLSTxx 18, 222
PROGxx 18
SMFPRMxx 18
System Measurement Facility
See SMF
time-in-system 113, 114, 178
definition of 1
transaction 3, 5, 10, 112, 113, 114, 138
TPPCB
See PCB
transaction
flow 10, 13
number of checkpoints 64
profile 2, 6, 69, 74, 76, 78, 79
SCHD parameter 213
transaction profile 1—3, 6—8, 69, 116, 198, 287, 361, 397
See also base profile
TSO 5, 49, 155
catalog 29
data set placement 134
dispatching priority 42, 294
swap paging 56
swap rates 141
swapping 57
users 234
work data sets 128
tuning overview 8
type 2 index (DB2) 334
U
UHTS 27
V
VAUT 25
Virtual Fetch
changing programs 231
description 229
IMS specification 230
IMS use 230
program load comparison 231
Virtual storage tuning tips 155
VSAM buffer pool 24
CI sizes 240
considerations 241
tuning 240
VSAM CI/CA splits 128
VSAM insert option 24
VSAM Strings 24
VSAMPL 24, 154, 155
VTAM 25, 29, 30, 38, 155
buffers 155
core 29, 36
dispatching priority 118, 119
LLAVLF 18
LU 6.2 297
node name 99
RECON 30
RECON buffers 98
VTAM (continued)
  SEND/RECEIVE macros  25
  tuning  29
  VAUT  25
  VTAMLST  18
  VTAMLST  18

W
WADS  31—34, 354
  data set placement  134
  dual  33
  example  244
  I/O time  121
Wait-for-input
  See WFI
WFI  114—115, 158, 208—334, 371
  dedicated region  192, 226
  occupancy  111
  process limit count  212
  processing limit count  64
  Schedule End to First DL/I Call  81
  storage isolation  159
  transactions  209
WKAP pool  89, 97
work load manager (WLM)
  goal mode  39, 234
  overview  39
  performance block  39
  used with IMS  39
worksheets  40, 43, 50, 69, 377, 386
  CPU Resource Worksheet  377
  I/O Resource Worksheet  379
  IMS Monitor  69
  IMS Resource Worksheet  383
  Real Storage Resource Worksheet  381
  RMF  40, 43, 50

X
XDFLD  287
XRF  34
ITSO Redbook Evaluation

IMS/ESA Version 5 Performance Guide
SG24-4637-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. Please complete this questionnaire and return it using one of the following methods:

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale: (1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

What other redbooks would you like to see published?

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

© Copyright IBM Corp. 1997