John Burgess
Arndt Eade

IBM CICS Performance Series: CICS and VSAM RLS

Introduction

The CICS® Performance Series is a collection of IBM® Redpaper™ publications that are focused on the performance of customer information control systems (CICS) and written by members of the IBM Hursley CICS development community. The topics are chosen based on feedback from CICS customers, with the goal of expanding readers' understanding of the rich features that are offered by the CICS Transaction Server (CICS TS) product portfolio.

This paper is the third in the series, following these earlier publications:

- *IBM CICS Performance Series: CICS TS V4.2 and Java Performance*, REDP-4850
  http://www.redbooks.ibm.com/abstracts/redp4850.html?Open
- *IBM CICS Performance Series: CICS, DB2, and Thread Safety*, REDP-4860
  http://www.redbooks.ibm.com/abstracts/redp4860.html?Open

The objectives of this publication, *CICS and VSAM RLS*, are threefold:

- Show comparisons and performance characteristics of a simple workload accessing VSAM data through the following mechanisms:
  - Locally defined Local Shared Resource (LSR) VSAM files
  - Remotely defined files accessed using Cross-Memory (XM) MRO Function Shipping
  - Remotely defined files accessed using XCF MRO Function Shipping
  - Locally defined Record Level Sharing (RLS) opened VSAM files
- Outline and explain the CICS and RLS activities that occur when a CICS program issues various API calls to RLS, specifically READ, READ UPDATE, REWRITE, ADD, BROWSE, and DELETE. This is viewed mainly from a performance perspective, but also gives readers a better understanding of the processing that occurs when an RLS call is made.
- Document how monitoring data that is generated by CICS, RMF™, and SMS can be extracted and correlated, and explain some of the key fields that you can examine to determine whether configuration or usage patterns can be optimized.

To achieve these objectives, this paper provides an overview of CICS and RLS concepts, terms, and components. It then follows up with some specific CICS migration scenarios that explain why customers should consider migrating to SMS-managed RLS data sets.

To fully understand the SMF 42 records that are produced by RLS, it is necessary to use a formatting program, the sample output of which is included later in this paper. The program, in object code format, is included in SupportPac CP13. Instructions on link editing and running the program can be found at the following website:

http://www-01.ibm.com/support/docview.wss?uid=swg24026507

CICS and RLS: Concepts, terms, and components

Before getting into the technical details, a quick review of CICS and RLS basics is in order.

RLS

VSAM RLS is a function first provided by DFSMS/MVS™ Version 1.3 and supported by CICS. It enables VSAM data to be shared, with full update capability, among many applications running in many CICS regions. With RLS, CICS regions that share VSAM data sets can reside in one or more MVS images within an MVS parallel sysplex.

In DFSMS/MVS Version 1.3, support was implemented for a new data sharing subsystem, SMSVSAM, which runs in its own address space. SMSVSAM provides the VSAM RLS support that is required by CICS application-owning regions (AORs) and batch jobs within each MVS system image in a parallel sysplex environment.

The SMSVSAM subsystem, which is generally initialized automatically during an MVS initial program load, uses the coupling facility for its cache and lock structures. It also supports a common set of buffer pools for each MVS image.

CICS and RLS

Before RLS, CICS users were able to share VSAM data sets with integrity by using function shipping to a file-owning region. With function shipping, one CICS region accesses the VSAM data set on behalf of other CICS regions. Requests to access the data set are shipped from the region where the transaction is running to the region that has access to the file.

Function shipping provides a solution for the CICS user, but it has limitations. For example, function shipping does not address the problems of sharing data sets between CICS regions and batch jobs.

It should also be noted that the FOR is constrained to a single TCB and, therefore, the speed of a single central processor (CP). RLS is not subject to this constraint because all of the work is done across multiple application-owning regions where the application resides. Like LSR, these RLS requests are capable of running in threadsafe mode, in which case they run on L8/9 TCBs. In non-threadsafe mode, the RLS requests run on SRBs within the CICS address spaces.

Data sets opened for RLS processing have the following characteristics to ensure data integrity:

- Buffer Coherency: RLS uses the buffer registration and invalidation functions of the coupling facility (CF) cache as the means to maintain buffer coherency across the local

buffer pools. RLS uses the conditional write function of the CF cache to implement an optimistic serialization protocol for changing data control intervals.

- Locking: RLS locking is performed at the level of individual records, which permits multiple concurrently executing transactions to change different records residing within the same data control interval (CI). RLS assigns a separate buffer containing a copy of the data CI to each of the sharers. The invalidate and conditional write functions of the CF local cache are used to detect concurrent write activity at the CI level. If the first write is successful, it invalidates the buffers assigned to the other sharers and causes their subsequent attempts to write to their buffers to fail. When RLS detects this failure, it internally reaccesses the data set and reapplies the change. This process merges the change with the earlier changes and is known as *record merge redo*.
- DASD write serialization: RLS uses the castout lock functions of the coupling facility to achieve this serialization.
- Control interval and control area split serialization: RLS uses an exclusive lock to serialize CI and CA processing for a key-sequenced data set (KSDS). RLS uses the CIDFBUSY flag, a private serialization mechanism of the VSAM KSDS architecture, to serialize requests to access data CIs while they are being split or moved by a control area split.

Figure 1 shows a typical configuration where CICS regions that are running in multiple LPARs of a sysplex are accessing files using function shipping. The CICS region that owns and accesses the files is commonly known as the *file-owning region* (FOR), while the CICS region that hosts the application logic and invokes function shipping requests (to access the files) is known as the *application-owning region* (AOR).

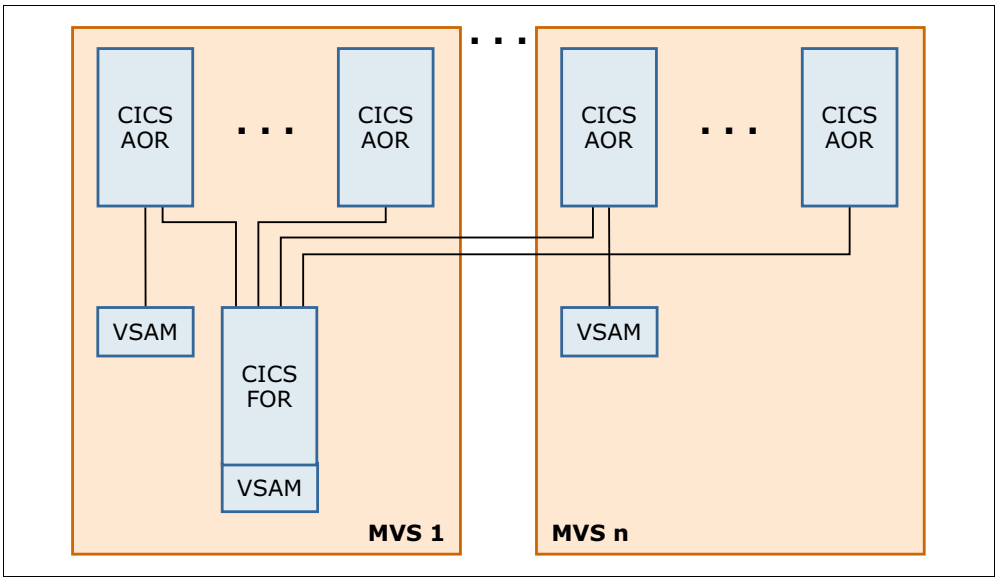

Figure 1 Sample CICS function shipping configuration

Figure 2shows the same configuration where the AORs have been defined to access the files directly using VSAM RLS. Note that the FOR is no longer required.

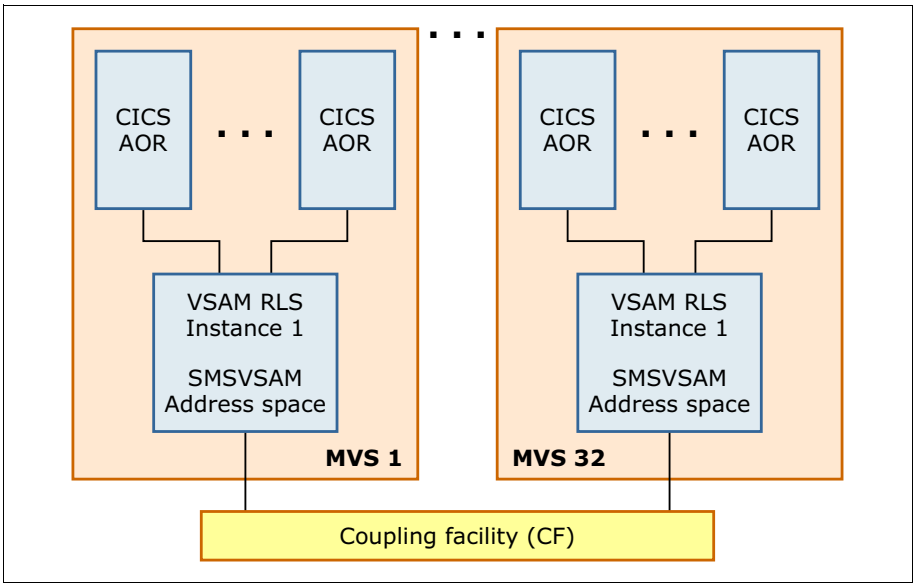

Figure 2 Sample CICS RLS configuration

Components of RLS

To coordinate and control access to VSAM files at a record sharing level, DFSMS, through the SMSVSAM address space, uses a number of components and facilities.

Each LPAR where RLS access is required will have an active SMSVSAM address space. This address space manages all access to data sets that are opened in RLS mode. To achieve this, SMSVSAM communicates by using XCF groups and CF cache and lock structures with SMSVSAM address spaces running on other LPARs within the sysplex.

In order for a VSAM data set to be opened for RLS access, it must be SMS-managed. The process of defining the data set as SMS-managed requires, among other things, a storage class to be assigned. The storage class definition will contain a cacheset name with a number of CF cache structures that are assigned to the named cacheset.

In addition to the cache structures used by SMSVSAM as intermediate storage between local memory and DASD, a lock structure, called IGWLOCK00, is used to provide sysplex-wide locking at the record level. In IBM z/OS® v1R10, up to 10 additional lock structures, called secondary lock structures, can now be defined. These lock structures, which contain only record locks, will provide better separation of workloads, CF balancing, and availability. Correct sizing of the CF cache and lock structures used by SMSVSAM is key to achieving optimum performance. For recommendations on correctly sizing these structures, refer to *z/OS DFSMStvs Planning and Operating Guide* (SC26-7348) in the z/OS Information Center at this address:

`http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.idat100/toc.htm`

Figure 3shows a sysplex with a typical CICS and SMSVSAM configuration.

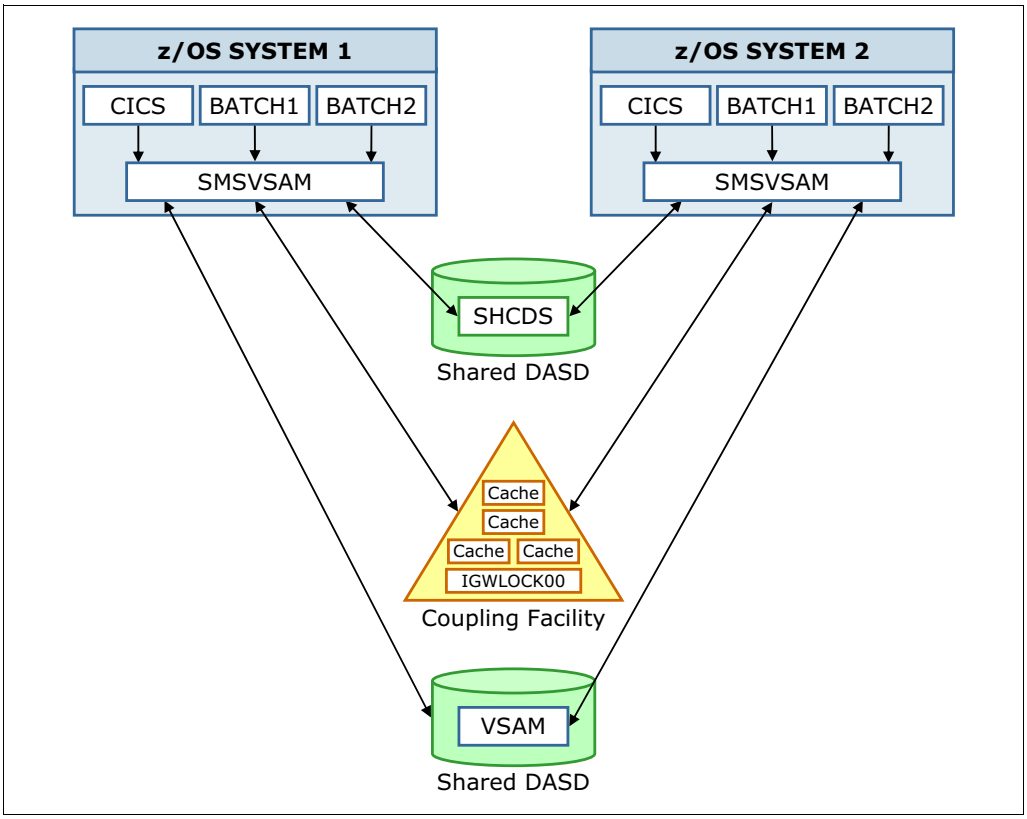

Figure 3 CICS and SMSVSAM sysplex configuration

File access processing

This section outlines the processing that occurs for each type of RLS file access that can be made from a CICS application program through a CICS API. For each API call, the possible paths are described along with any factors that can significantly affect the processing that occurs.

EXEC CICS READ and BROWSE (READNEXT and READPREV)

A single read, whether as a result of an EXEC CICS READ or an EXEC CICS READNEXT API call, results in the same underlying processing occurring. The processing involves the following steps:

1. A cross memory call is made from CICS to SMSVSAM requesting the record.
2. SMSVSAM first checks its local buffer pool for the record. If the record is within a local buffer pool and still valid (see the note that follows), then the data is returned and the request is satisfied. If the record is not found in a local buffer pool, SMSVSAM next checks the CF cache structure associated with the data set. If the record can be located there, it is read into the local buffer pool and returned to CICS. If the record is not found there, SMSVSAM performs an I/O request to DASD to locate the CI, registers an interest in the CI, puts the CI in the CF cache structure, and reads the records into its buffer pool.
3. The locks obtained and held during the read operation depend upon the READINTEG setting of the request. This is discussed in more detail later in this document.

Important: If SMSVSAM on a different LPAR has updated the same CI, then the version that is held in the local SMSVSAM buffer pools becomes invalid. In this event, the new valid version must be re-read into the buffer pools from the CF cache structure prior to the record being returned to the application.

EXEC CICS READ UPDATE

The processing for an EXEC CICS READ UPDATE API request is essentially the same as that for an EXEC CICS READ API request, with the exception that a record lock is requested and held until syncpoint for files that are recoverable.

EXEC CICS REWRITE

In order to rewrite a record, SMSVSAM will have obtained an exclusive lock on the record in the IGWLOCK00 structure as part of the EXEC CICS READ UPDATE API request.

When the rewrite request is issued, the CI containing the updated record is updated in the local buffer pool and then in both the CF cache structure and DASD. During the write to DASD, the castout lock will be held in the CI.

Any SMSVSAM address space on a foreign LPAR will now have its copy of the CI that contains the updated record marked as invalid (*cross-invalidated)* and will need to refresh the buffer pool copy from the CF cache.

At syncpoint, the exclusive record lock is released.

Updates to different records in the same CI

If two CICS regions on different LPARs within the sysplex are updating different records in the same CI at the same time, the processing occurs as follows:

1. CICS1 on MVS1 reads Record A from CI1 at the same time that CICS2 on MVS2 reads Record B from CI1.
2. CICS2 does its rewrite operation first. During the rewrite to DASD, CICS2 holds a castout lock for the CI.
3. CICS1 tries to rewrite its version of the CI but is held up because it cannot get the CASTOUT lock held by CICS1. It retries the rewrite in 1.5 milliseconds. If APAR OA30499 is installed, this retry interval will be self-tuning based on DASD response times.
4. When CICS1 does get to perform the rewrite, it ascertains that the CI has changed and that the changes made by CICS1 must be merged back into the CI. This is performed in the background and is transparent to the application.

   The process of reapplying the record update to the CI is called a *redo*. A count of redo operations is recorded in the SMF type 42 records that are generated by SMSVSAM.

Setting up CICS to exploit RLS

The ability to access VSAM files using RLS is an integral part of CICS Transaction Server. It means that any CICS application that is currently accessing VSAM files locally, whether using LSR or by function shipping to an FOR region, can access the same VSAM files in RLS mode.

No modification or recompilation of the application program is required. You simply need to alter the CICS file definition as described in the next section and specify **RLS=YES** in the CICS System Initialization Table (SIT).

CICS file definitions

The CICS resource definition for files is described in the CICS Transaction Server for z/OS V4.2 Information Center at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp

For the CICS file definition, the RLSACCESS attribute determines whether the file will be opened in RLS mode.

Another important attribute that can significantly affect performance is the CICS FILE attribute READINTEG. The default setting of UNCOMMITTED will result in the same data integrity as CICS Local Shared Resource (LSR) access. Any other setting will result in an increase in locking activity for read requests.

Consult the available CICS and DFSMS resources to learn how each possible setting will affect READ requests. In addition to the CICS Transaction Server for z/OS V4.2 Information Center referenced previously, there is a z/OS DFSMS section in the z/OS V1R13 Information Center available at this address:

http://publib.boulder.ibm.com/infocenter/zos/v1r13/index.jsp?topic=%2Fcom.ibm.zos.r13.ida%2Fida.htm

CICS program definitions

While attributes of the program have little effect on RLS file access, there is one parameter that is worth mentioning with respect to RLS: CONCURRENCY.

The CONCURRENCY attribute determines whether the program runs under the Quasi-reentrant (QR) TCB (when QUASIRENT is specified) or under an Open Transaction Environment (OTE) TCB (when THREADSAFE is specified).

In the first case, RLS file access requests are executed on dedicated SRBs, while in the second case, the RLS access request is performed on OTE L8 or L9 TCBs. For performance measurement purposes, the SRB CPU usage is stored in the RLSCPUT field and is an accurate reflection of time consumed performing RLS activities. For the second case, L8 CPU consumption is for all activity that is performed on the L8, which means that CPU usage solely for RLS processing cannot be determined.

Migration scenarios

To show the effect of migrating between configurations, some simple in-house benchmarks were run in various scenarios:

- Scenario 1 involved a terminal owning region (TOR) and an application owning region (AOR) where all the files were accessed locally in the AORs by means of LSR pools.
- Scenario 2 introduced a file owning region (FOR) and function-shipped all of the file requests through MRO.
- Scenario 3 moved the FOR to another LPAR in the sysplex so that all of the file requests could be function-shipped across XCF using a coupling facility.
- Scenario 4 removed the FOR and redefined all of the files in the AORs as RLS files so that all of the file requests were now RLS accesses through the SMSVSAM address space. The files are defined with the default read integrity of Uncommitted (UR).
- Scenario 5 is the same configuration as scenario 4 with the file read integrity changed to Consistent Read (CR).

- Scenario 6 is the same configuration as scenario 4 with the file read integrity changed to Repeatable Read (RR).

The three RLS access types mentioned have the following characteristics:

- Uncommitted (UR) (No Read Integrity): In this case, there are no Lock accesses for read requests. An EXEC CICS READ API request will see records that might have already been Read for Update by another application and are about to be changed with an EXEC CICS REWRITE API request. This is the same as the situation for CICS API accesses to LSR files.
- Consistent Read (CR): When a record has been read for update, no other transaction can read or browse the same record until it has been committed with a syncpoint. A Consistent Read has to obtain a shared lock, but this cannot occur if another transaction has an exclusive lock obtained as part of an EXEC CICS READ UPDATE API request.
- Repeatable Read (RR): When a record has been read for update, no other transaction can read or browse the same record until it has been committed. This is the same as Consistent Read, except that when another transaction has read a record, the shared lock that is obtained is held until syncpoint, so no other transaction can Read for Update this record during this time.

Scenario setup

Numerous configuration and setup factors influenced the implementation of the scenarios.

Sysplex configuration

The sysplex consisted of two LPARs, each with three dedicated CPs on an IBM 2097-763 Model E64. There were two integrated CFs with three shared CPs each.

CICS configurations

CICS Transaction Server Version 4.1 was used, running on z/OS Version 1.11. Network simulators were used to drive the workload at approximately the same transaction rates for each scenario. All transactions entered the TOR and were routed to the AOR, where file requests were either function-shipped or accessed locally as LSR or RLS files.

Workload

The workload consisted of COBOL applications with little business logic. The applications were not defined as threadsafe. The workload had the following characteristics:

- Average of six file requests per transaction
- CICS SIT parameter MROLRM=YES
- CICS file requests: 69% Read, 10% Read for Update, 9% Update, 11% Add, 1% Delete
- CICS SIT parameter FCQRONLY=YES
- LOG(UNDO) attribute specified on the VSAM Cluster definitions

The RLS buffer pool size, CF caches, and so on, were sized to avoid any major performance constraints in the system.

FCQRONLY=YES will give slightly better performance in a CICS region where you know that there are no threadsafe applications. If applications are running in threadsafe mode, this must be set to NO or else all file control requests will switch to the QR TCB for processing.

Measurement methodology

Performance data was collected for five distinct transaction throughput rates. Network simulators entered transactions in the TORs using a particular *think time* (a term used to

describe the delay between the arrival of data at the terminal and the subsequent user response). The workload was allowed to reach steady state and then performance data was collected using RMF Monitor I. The think time was then reduced to increase throughput, and this process was repeated for the five different throughput rates. The tables presented here show only the RMF Monitor I data for the workload, but examples of the other data collected are shown elsewhere in this paper.

Scenario results

The results of each scenario are presented here.

Understanding the tables

The scenario results tables show data extracted from RMF Monitor I during the measurement intervals. The columns are organized as follows:

ETR	Transactions per second counted at the TOR
TOR%	Percentage of one CP used by the TOR during the interval
AOR%	Percentage of one CP used by the AOR during the interval
FOR%	Percentage of one CP used by the FOR during the interval
TOT. CPU%	Sum of TOR%, AOR% and FOR%
CPU/TRAN	Sum of address space CPU divided by the transaction rate
RESP Time	Total transaction response time measured in the TOR

Also shown, when appropriate, is the CPU percentage for XCF address space (XCFAS) on both the LPARs and the SMSVSAM address space.

Scenario 1 (Local LSR access in the AOR)

Table 1 displays the results for the first scenario involving a TOR and AOR where all of the files were accessed locally in the AORs by means of LSR pools.

Table 1 Local LSR access in the AOR

ETR	TOR%	AOR%	TOT. CPU%	CPU/TRAN	RESP Time
423.46	3.79	11.35	15.14	0.357 ms	7 ms
531.61	4.72	14.07	18.79	0.353 ms	7 ms
702.08	6.14	18.24	24.38	0.347 ms	7 ms
1041.40	9.03	27.04	36.07	0.346 ms	7 ms

Using the data recorded here, the average CPU time per transaction across the five intervals can be calculated to be 0.34 ms.

Scenario 2 (MRO function shipping)

Table 2 provides the results for the second scenario, which introduced an FOR and function-shipped all of the file requests through MRO.

Table 2 MRO function shipping

ETR	TOR%	AOR%	FOR%	TOT. CPU%	CPU/TRAN	RESP time
427.41	4.09	10.68	6.42	21.19	0.495 ms	11 ms
536.87	5.05	13.34	7.79	26.18	0.487 ms	11 ms
711.31	6.51	17.36	9.91	33.78	0.474 ms	11 ms
1065.26	9.52	25.57	14.67	49.76	0.467 ms	11 ms
1426.78	12.56	33.54	19.04	65.14	0.456 ms	11 ms

After migrating to MRO function shipping, the average CPU time per transaction has now gone up to 0.47 ms. Yet because these applications have little business logic, you should not view this increase in terms of percentage. Instead, take the absolute deltas and apply them to the average of six function-ships per transaction: 0.13 ms divided by 6, or 0.021 ms per function-ship. This number can then be scaled from these IBM 2097 703 CPU times to another processor using the LSPR and applied to other application profiles to get a rough estimate of the increase for a particular application.

This same methodology can be used for any of these measurement comparisons.

Scenario 3 (XCF function shipping)

Table 3 shows the results of the third scenario, which moved the FOR to another LPAR in the sysplex so that all of the file requests could be function-shipped across XCF using a coupling facility.

Table 3 XCF function shipping

ETR	TOR%	AOR%	XCFAS%	FOR%	XCFAS%	TOT. CPU%	CPU/TRAN	RESP time
426.75	3.93	13.43	3.13	8.30	3.54	32.33	0.757 ms	12 ms
539.29	4.93	16.90	4.05	10.41	4.42	40.71	0.754 ms	12 ms
720.57	6.52	22.49	5.48	13.75	5.92	54.16	0.751 ms	12 ms
1072.64	9.69	33.42	8.17	20.04	8.92	80.24	0.748 ms	12 ms
1435.46	13.09	44.8	11.2	27.06	11.98	108.13	0.753 ms	12 ms

By moving the FOR to another LPAR and then function shipping using the coupling facility, CPU usage was introduced in the XCFAS address spaces on each LPAR. Now the average CPU per transaction has gone up to 0.75 ms. Using the average of six function-ships per transaction, the cost per XCF function-ship can be calculated.

Scenario 4 (RLS Uncommitted Read)

The results for the fourth scenario are shown in Table 4. The FOR was removed and all of the files in the AORs were redefined as RLS files, meaning that all file requests were now RLS accesses through the SMSVSAM address space.

Table 4 RLS Uncommitted Read

ETR	TOR%	AOR%	SMSVSAM%	TOT. CPU%	CPU/TRAN	RESP Time
423.50	3.93	17.50	0.48	21.87	0.515 ms	4 ms
532.29	4.83	21.46	0.55	26.84	0.504 ms	4 ms
703.15	6.30	28.08	0.66	35.04	0.498 ms	4 ms
1042.72	9.12	40.91	0.90	50.93	0.488 ms	4 ms
1378.79	12.07	54.11	1.17	67.35	0.488 ms	4 ms

Notes on Uncommitted Read (UR):

- This has the same characteristics as LSR files in that it is possible to read a record that has been read for update by another task, which could change it with a rewrite.
- In this scenario, we removed the FOR from the configuration and defined all of the files in the AORs as RLS with Uncommitted Read (UR) integrity. If we were comparing this to XCF function shipping, then, as the data shows, the CPU per transaction has gone down to an average of 0.49 ms.

Scenario 5 (RLS Consistent Read)

Table 5 shows the results for the fifth scenario. It was identical to scenario four with the exception that the read integrity of the files was changed from Uncommitted Read (UR) to Consistent Read (CR).

Table 5 RLS Consistent Read

ETR	TOR%	AOR%	SMSVSAM%	TOT. CPU%	CPU/TRAN	RESP Time
423.64	4.00	20.63	0.50	25.13	0.593 ms	5 ms
532.09	5.00	25.49	0.56	31.05	0.583 ms	5 ms
703.11	6.46	33.00	0.69	40.15	0.571 ms	5 ms
1043.03	9.80	50.75	0.95	61.50	0.589 ms	5 ms
1378.54	12.32	64.26	1.22	77.80	0.564 ms	5 ms

Notes on Consistent Read (CR):

- When a record has been read for update by a task, no other transaction can read or browse the same record until it has been committed with a syncpoint.
- A consistent read has to obtain a shared lock, but this cannot be done while another task has an exclusive lock obtained as part of a read for update API request.
- The shared lock is obtained and released as soon as the record is read. This locking costs extra CPU time and additional accesses to the coupling facility.

The average CPU time per transaction was now 0.58 ms.

Scenario 6 (RLS Repeatable Read)

Table 6 shows the results for the sixth scenario, It was identical to Scenario 4 with the exception that the read integrity of the files was changed from Uncommitted Read (UR) to Repeatable Read (RR).

Table 6 RLS repeatable Read

ETR	TOR%	AOR%	SMSVSAM%	TOT. CPU%	CPU/TRAN	RESP Time
423.64	4.07	22.09	0.49	26.65	0.629 ms	5 ms
532.04	5.17	24.16	0.56	29.89	0.561 ms	5 ms
703.11	6.65	31.23	0.66	38.54	0.548 ms	5 ms
1043.03	9.49	45.17	0.92	55.58	0.532 ms	5 ms
1378.54	12.42	59.30	1.22	72.94	0.529 ms	5 ms

Notes on Repeatable Read (RR):

- When a record has been read for update, no other transaction can read or browse the same record until it has been committed. This is identical to the Consistent Read (CR) scenario except that when another task has read a record, the shared lock that is obtained is held until the reader has syncpointed, which means that no other transaction can read for update this record during this time.
- At the higher transaction rates, the CPU time per transaction was slightly less than in the Consistent Read (CR) scenario. This was mostly due to fewer accesses to the CF for locks.
- When using Repeatable Read (RR), only the first read in a transaction for a particular record will obtain a lock. Multiple reads for the same record will not need multiple lock accesses, so depending on access patterns, Repeatable Read could use less CPU time per transaction but can restrict throughput.

Summary of performance measurements

The scenario results show that the best performance, in terms of CPU cost per transaction, comes from local LSR files. When data is made available for sharing (using function shipping or RLS), the cost increases.

The net effect of migrating to RLS will depend upon the original configuration:

- If the migration is from MRO, with a high proportion of requests being function-shipped across XCF links, then there could be a reduction in overall CPU cost per transaction.
- If the migration is from MRO/XM files or local files, then there will be an increase in CPU cost per transaction.

The workload showed approximately 5% increase in CPU cost per transaction when migrating from MRO/XM function shipping to RLS. Other workloads will vary depending on the path length of the application and the number of file requests per transaction.

RLS has better scaling capabilities than CICS function shipping because it is not limited to a single FOR that is constrained to the speed of a CP due to its single-TCB architecture.

Monitoring the system

This section describes the data that can be collected to help analyze the performance of CICS-RLS systems. It shows how to start collecting the data and how it can be post-processed, and explains some of the key fields in the data.

When analyzing the performance of CICS-RLS applications, data can be collected from the following sources:

- CICS:
  - CICS Statistics: SMF 110 records, subtype 2
  - CICS Monitoring: SMF 110 records, subtype 1
- RMF Monitor I and Monitor III:
  - SMF 70-79 records
  - Online interactive displays with Monitor III through the **RMFWDM TSO** command.
- SMSVSAM:
  - SMF 42 records, subtypes 15, 16, 17 18, and 19

CICS data

This section introduces and explains some of the tools and techniques used to monitor CICS applications using RLS.

CICS Statistics

CICS Statistics can be used to determine the types of file accesses and the rates. They can also be used to determine the CPU usage on all TCBs. Non-threadsafe applications run on the QR TCB, which will be constrained to the speed of a single CP. RLS requests in this mode run on SRBs in the CICS address space. Threadsafe applications run concurrently on L8 or L9 TCBs, where the RLS request also runs.

CICS Statistics are only really useful when the interval they cover is relatively short and the counters have been reset at the start of the interval (that is, when you are ready to start collecting data for a measurement period). This can be done in either of two ways:

- Use CEMT to reset the statistics and then, after the required time, use CEMT to perform a statistics collection.
- Use CEMT to set interval statistic recording to the required time interval. When CICS interval statistics are written, the counters are automatically reset. This method also allows you to set the end-of-day time so you can synchronize collection with RMF and SMSVSAM recording.

Example 1is the output from a **CICS CEMT INQ STAT** command that was executed at 12:11. The interval is set to 15 minutes and the end-of-day is set to 12:15, so there will be a short interval of 4 minutes and then, after that, all 15-minute intervals will be on the quarter-hour.

Example 1 Sample output from CICS CEMT INQ STATISTICS command

```
CEMT INQ STAT
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Sta On          Int( 001500 ) End( 121500 )
Nex(121500
```

The CICS Statistics formatting program, DFHSTUP, can be used to format the CICS Statistics records. Example 2 shows sample output from a formatted statistics report, including the file-access counts and types.

Example 2 Extract from DFHSTUP report

```
FILES - Requests Information
+_____________________________
0  File      Get       Get Upd  Browse    Update    Add       Delete    Brws Upd VSAM EXCP Requests
   Name      Requests  Requests Requests  Requests  Requests  Requests  Requests    Data     Index
+  ____________________________________________________________________________________________
   ACCUNTDB    12710         0         0         0         0         0         0     12710     12710
   ACCUNTDX    12425         0         0         0         0         0         0     12425     12425
   COMPOSDB    27864         0         0         0         0         0         0     27864     27864
   COMPOSDX    28566         0         0         0         0         0         0     28566     28566
   CUSTOMER     4707      1572         0      1572      6276         0         0     17093     25193
   CUSTOMEX     4644      1547         0      1547      6218         0         0     16985     24915
```

For RLS, VSAM Execute Channel Program (EXCP) requests is a count of the number of calls to the system buffer manager (unlike with LSR, where it is the number of EXCPs). The count includes calls that result in either a CF cache access or an I/O request.

Example 3is an extract from the Dispatcher section of CICS Statistics showing the total address space CPU time since the last reset on the TCBs and SRBs within the CICS address space.

Example 3 Extract from Dispatcher section of DFHSTUP report

```
Dispatcher Start Date and Time. . . . . . . : 08/27/2009  09:23:32.3348
Address Space CPU Time. . . . . . . . . . . :     00:00:28.624146
Address Space SRB Time. . . . . . . . . . . :     00:00:20.033857
```

In this case, the SRB activity is relatively high, which indicates that the application is not threadsafe and so the RLS requests are not handled on open TCBs but on SRBs.

CICS Monitoring

CICS Monitoring will generate an SMF 110 record (subtype 1) for every transaction, or if required for long-running transactions, at every syncpoint. Monitoring can be turned on using the CICS System Initialization Table (SIT) parameter MNPER=ON, or dynamically using CEMT.

If you want to collect information for only a 15-minute interval and coordinate it with CICS statistics, then execute the CEMT SET MON PERFCLASS(PERF) command to activate collection at the start of the interval and use the CEMT SET MON NOPERF command to deactivate collection at the end of the interval. CICS Performance Monitoring can generate large numbers of SMF records, so if it is always turned on, you must put processes in place to ensure correct sizing and offloading of the SMF datasets.

CICS Monitoring records can be processed using CICS Performance Analyzer (PA). The field names in Table 7are of interest when looking at RLS performance.

Table 7 CMF and PA field names for file accesses

CICS CMF name	CICS PA name	Description	CICS API requests
RLSCPUT	RLSCPU	RLS file request CPU (SRB) time	CPU (SRB) time used to perform RLS request
RLSWAIT	RLSWAIT	File I/O wait	RLS file I/O wait time
FCGETCT	FCGET	File GET requests	READ and READ UPDATE
FCPUTCT	FCPUT	File PUT requests	REWRITE

CICS CMF name	CICS PA name	Description	CICS API requests
FCBRWCT	FCBROWSE	File Browse requests	READNEXT and READPREV
FCADDCT	FCADD	File ADD requests	WRITE
FCDELCT	FCDELETE	File DELETE requests	DELETE
FCTOTCT	FCTOTAL	Total file requests	Above plus STARTBR and ENDBR

CICS PA can generate many types of reports. Example 4 is an example of a summary report showing some of the data relevant to RLS applications.

Example 4 CICS Performance Analyzer summary report of file accesses

Tran	#Tasks	Avg User CPU Time	Avg RLS CPU Time	Avg RLS Wait Time	Avg FCGET Count	Avg FCPUT Count	Avg FCADD Count	Avg FCBROWSE Count	Avg FCDELETE Count	Avg FC Total Count	Avg Response Time
DE29	271	.0014	.0004	.0003	8	0	2	0	0	10	.0244
DX1	2469	.0013	.0002	.0026	2	1	0	0	0	3	.0441
DX20	237	.0014	.0004	.0001	8	0	2	0	0	10	.0242
DX29	266	.0014	.0004	.0001	8	0	2	0	0	10	.0237
IT2	1791	.0014	.0003	.0000	17	0	0	0	0	17	.0212
IT8	2483	.0015	.0005	.0045	9	2	3	0	0	14	.0707
IX2	1871	.0014	.0003	.0000	17	0	0	0	0	17	.0211
IX8	2456	.0014	.0005	.0046	9	2	3	0	0	14	.0713
OE1	1560	.0012	.0001	.0000	3	0	0	0	0	3	.0208
OE2	1557	.0013	.0002	.0028	1	1	1	0	0	3	.0587
OE4	1558	.0014	.0001	.0000	9	0	0	0	0	9	.0208

This extract shows CPU time, response time, and file access counts. The RLS CPU time is the CPU time that RLS used to complete the request on the SRB. If the application were threadsafe, all of the application time and RLS time would be accounted for on L8 or L9 TCBs and not SRBs. In this case, RLS CPU time would be reported as zero.

If you want to run a CICS PA file summary report that shows how many file accesses have been performed by each transaction, you need to specify MNRES=ON in the CICS SIT. Note that if you do not make changes to your CICS Monitoring Control Table (MCT), by default it will keep track of only eight files per transaction.

Standard CICS performance monitoring class data includes totals for *all* of the files accessed by a transaction. Transaction resource monitoring, in contrast, collects information about individual files, up to the number specified in the MCT (`DFHMCT TYPE= INITIAL,FILE=n`).

CICS Performance Analyzer reports

There are a number of CICS Performance Analyzer reports that show key aspects of transactions, including:

- Performance TOTAL
- Wait Analysis
- Resource Usage

Each report is explained briefly here. For a more detailed explanation of the reports available in CICS Performance Analyzer, consult the CICS Transaction Server for z/OS Information Center at this website:

`http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp`

Performance TOTAL report

The Performance TOTAL report summarizes all activity performed by a transaction or transactions.

Example 5 shows the statements used to generate the CICS Performance TOTAL report.

Example 5 Sample control statements for CICS PA Performance TOTAL report

```
CICSPA IN(SMFIN001),
       APPLID(<applid>),
       LINECNT(60),
       FORMAT(':','/'),
       PRECISION(4),
       TOTAL(OUTPUT(<ddname>),
             TITLE1('<title>'),
             SELECT(PERFORMANCE(
             INC(TRAN(<transaction identifier>),
                 START(FROM(,00:00:00.00),
                         TO(,23:59:59.99)),
                  STOP(FROM(,00:00:00.00),
                         TO(,23:59:59.99))))))
```

Example 6 shows the output from a sample CICS Performance TOTAL report. It provides a detailed breakdown of all the fields in the CMF performance class records.

Example 6 Sample output from CICS Performance TOTAL report

```
                                 Dispatched Time          CPU Time
                                DD HH:MM:SS    Secs   DD HH:MM:SS    Secs

Total Elapsed Run Time            00:20:00    1200

From Selected Performance Records

QR Dispatch/CPU Time              00:00:08       8     00:00:04       4
MS Dispatch/CPU Time              00:00:00       0     00:00:00       0
                                  --------   -----     --------  ------
 TOTAL (QR + MS)                  00:00:08       8     00:00:04       4

L8 CPU Time                                           00:00:00       0
J8 CPU Time                                           00:00:00       0
S8 CPU Time                                           00:00:00       0
X8 CPU Time                                           00:00:00       0
                                  --------   -----     --------  ------
 TOTAL (L8 + J8 + S8 + X8)        00:00:00       0     00:00:00       0

L9 CPU Time                                           00:00:00       0
J9 CPU Time                                           00:00:00       0
X9 CPU Time                                           00:00:00       0
                                  --------   -----     --------  ------
 TOTAL (L9 + J9 + X9)             00:00:00       0     00:00:00       0

                                  --------   -----     --------  ------
 Total CICS TCB Time              00:00:08       8     00:00:04       4



Total Performance Records (Type C)              0
Total Performance Records (Type D)              0
Total Performance Records (Type F)              0
Total Performance Records (Type S)              0
Total Performance Records (Type T)            302
                                           --------                                  --------
Total Performance Records (Selected)          302          Total Performance Records    20097
```

From Selected Performance Records	..... C O U N T S ..... Total	Avg/Task	Max/Task	..... T I M E ..... Total	Avg/Task	Max/Task
Dispatch Time	1823	6.0	235	8	.026	4.148
CPU Time				4	.012	1.899
RLS CPU (SRB) Time				3	.009	1.229
Suspend Time	1823	6.0	235	32	.104	11.152
Dispatch Wait Time	1521	5.0	234	5	.017	1.758
Dispatch Wait Time (QR Mode)	1521	5.0	234	5	.017	1.758
Response (-TCWait for Type C)				0	.000	.000
Response (All Selected Tasks)				39	.131	12.777
QR Dispatch Time	1823	6.0	235	8	.026	4.148
MS Dispatch Time	0	.0	0	0	.000	.000
RO Dispatch Time	0	.0	0	0	.000	.000
QR CPU Time				4	.012	1.899
MS CPU Time				0	.000	.000
RO CPU Time				0	.000	.000
L8 CPU Time				0	.000	.000
L9 CPU Time				0	.000	.000
J8 CPU Time				0	.000	.000
J9 CPU Time				0	.000	.000
S8 CPU Time				0	.000	.000
X8 CPU Time				0	.000	.000
X9 CPU Time				0	.000	.000

From Selected Performance Records	..... C O U N T S ..... Total	Avg/Task	Max/Task	..... T I M E ..... Total	Avg/Task	Max/Task
FCWAIT File I/O wait time	0	.0	0	0	.000	.000
RLSWAIT RLS File I/O wait time	715	2.4	40	16	.053	10.850
TSWAIT VSAM TS I/O wait time	0	.0	0	0	.000	.000
TSSHWAIT Asynchronous Shared TS wait time	125	.4	2	0	.001	.023
JCWAIT Journal I/O wait time	455	1.5	8	2	.007	.128
TDWAIT VSAM transient data I/O wait time	0	.0	0	0	.000	.000
IRWAIT MRO link wait time	3	.0	2	0	.000	.037
CFDTWAIT CF Data Table access requests wait time	0	.0	0	0	.000	.000
CFDTSYNC CF Data Table syncpoint wait time	0	.0	0	0	.000	.000
RUNTRWAI BTS run Process/Activity wait time	0	.0	0	0	.000	.000
SYNCDLY SYNCPOINT parent request wait time	0	.0	0	0	.000	.000
RMITIME Resource Manager Interface (RMI) elapsed time	1812	6.0	6	0	.000	.000
RMISUSP Resource Manager Interface (RMI) suspend time	0	.0	0	0	.000	.000
JVMITIME JVM initialize elapsed time	0	.0	0	0	.000	.000
JVMTIME JVM elapsed time	0	.0	0	0	.000	.000
JVMRTIME JVM reset elapsed time	0	.0	0	0	.000	.000
JVMSUSP JVM suspend time	0	.0	0	0	.000	.000
DB2CONWT DB2 Connection wait time	0	.0	0	0	.000	.000
DB2RDYQW DB2 Thread wait time	0	.0	0	0	.000	.000
DB2WAIT DB2 SQL/IFI wait time	0	.0	0	0	.000	.000
IMSWAIT IMS (DBCTL) wait time	0	.0	0	0	.000	.000
WMQGETWT WebSphere MQ GETWAIT wait time	0	.0	0	0	.000	.000
TCWAIT Terminal wait for input time	0	.0	0	0	.000	.000
LU61WAIT LU6.1 wait time	0	.0	0	0	.000	.000
LU62WAIT LU6.2 wait time	0	.0	0	0	.000	.000
SZWAIT FEPI services wait time	0	.0	0	0	.000	.000
SOWAIT Inbound Socket I/O wait time	0	.0	0	0	.000	.000
OSOWAIT Outbound Socket I/O Wait Time	0	.0	0	0	.000	.000
ISWAIT IPCONN link wait time	0	.0	0	0	.000	.000
RQRWAIT Request Receiver Wait Time	0	.0	0	0	.000	.000
RQPWAIT Request Processor Wait Time	0	.0	0	0	.000	.000
DSPDELAY First dispatch wait time	302	1.0	1	2	.008	.142
TCLDELAY First dispatch TCLSNAME wait time	0	.0	0	0	.000	.000
MXTDELAY First dispatch MXT wait time	0	.0	0	0	.000	.000
ENQDELAY Local Enqueue wait time	0	.0	0	0	.000	.000
GNQDELAY Global Enqueue wait time	0	.0	0	0	.000	.000

From Selected Performance Records	Total	Avg/Task	Max/Task	Total	Avg/Task	Max/Task
ICDELAY Interval Control (IC) wait time	8	.0	6	9	.030	7.039
GIVEUPWT Give up control wait time	212	.7	183	2	.005	1.423

```
WAITCICS CICS ECB wait time                                0        .0         0          0        .000        .000
WAITEXT  External ECB wait time                            0        .0         0          0        .000        .000
PTPWAIT  327O Bridge Partner wait time                     0        .0         0          0        .000        .000
RRMSWAIT Resource Recovery Services indoubt wait time      0        .0         0          0        .000        .000
LOCKDLAY Lock Manager (LM) wait time                       2        .0         2          0        .000        .010
DSTCBMWT Dispatcher TCB Mismatch wait time                 0        .0         0          0        .000        .000
MAXOTDLY Maximum Open TCB delay time                       0        .0         0          0        .000        .000
MAXJTDLY Maximum JVM TCB delay time                        0        .0         0          0        .000        .000
MAXSTDLY Maximum SSL TCB delay time                        0        .0         0          0        .000        .000
MAXXTDLY Maximum XPLink TCB delay time                     0        .0         0          0        .000        .000
DSMMSCWT DS storage constraint wait time                   0        .0         0          0        .000        .000
PCLOADTM Program Library wait time                         0        .0         0          0        .000        .000
SYNCTIME SYNCPOINT processing time                       302       1.0         1          1        .003        .125
OTSINDWT OTS Indoubt Wait time                             0        .0         0          0        .000        .000
EXWAIT   Exception Conditions wait time                    0        .0         0          0        .000        .000
DSCHMDLY Redispatch wait time caused by change-TCB mode    0        .0         0          0        .000        .000
TCMSGIN1 Messages received count                           0        .0         0
TCCHRIN1 Terminal characters received count                0        .0         0
TCMSGOU1 Messages sent count                               0        .0         0
TCCHROU1 Terminal characters sent count                    0        .0         0
TCMSGIN2 Messages received from LU6.1                      0        .0         0
TCCHRIN2 LU6.1 characters received count                   0        .0         0
TCMSGOU2 Messages sent to LU6.1                            0        .0         0
TCCHROU2 LU6.1 characters sent count                       0        .0         0
TCALLOC  TCTTE ALLOCATE requests                           0        .0         0
TCM62IN2 LU6.2 messages received count                     0        .0         0
TCC62IN2 LU6.2 characters received count                   0        .0         0
TCM62OU2 LU6.2 messages sent count                         0        .0         0
TCC62OU2 LU6.2 characters sent count                       0        .0         0
ISALLOC  Allocate Session requests for sessions on IP      0        .0         0
FCADD    File ADD requests                                 0        .0         0
FCBROWSE File Browse requests                            895       3.0       192
FCDELETE File DELETE requests                              4        .0         1
FCGET    File GET requests                             32798     108.6     18084

From Selected Performance Records                      Total   Avg/Task   Max/Task       Total    Avg/Task     Max/Task
FCPUT    File PUT requests                               185         .6          4
FCTOTAL  File Control requests                         34013      112.6      18128
FCAMCT   File access-method requests                   34189      113.2      18129
TDGET    Transient data GET requests                       0         .0          0
TDPUT    Transient data PUT requests                      10         .0         10
TDPURGE  Transient data PURGE requests                     0         .0          0
TDTOTAL  Transient data Total requests                    10         .0         10
TSGET    Temporary Storage GET requests                  604        2.0          2
TSPUTAUX Auxiliary TS PUT requests                         0         .0          0
TSPUTMAI Main TS PUT requests                              0         .0          0
TSTOTAL  TS Total requests                               604        2.0          2
BMSMAP   BMS MAP requests                                187         .6          1
BMSIN    BMS IN requests                                   0         .0          0
BMSOUT   BMS OUT requests                                252         .8          1
BMSTOTAL BMS Total requests                              439        1.5          2
JNLWRITE Journal write requests                            0         .0          0
LOGWRITE Log Stream write requests                       332        1.1          5
ICSTART  Interval Control START or INITIATE requests       0         .0          0
ICTOTAL  Interval Control requests                         8         .0          6
SC24CGET CDSA GETMAINs below 16MB                          0         .0          0
SC31CGET ECDSA GETMAINs above 16MB                         7         .0          1
SC24CHWM CDSA HWM below 16MB                               0         .0          0
SC31CHWM ECDSA HWM above 16MB                           3440       11.4        512
SC24COCC CDSA Storage Occupancy below 16MB                 0         .0          0
SC31COCC ECDSA Storage Occupancy above 16MB                0         .0          0
SC24UGET UDSA GETMAINs below 16MB                      32998      109.3      21716
SC31UGET EUDSA GETMAINs above 16MB                    136570      452.2      89975
SC24UHWM UDSA HWM below 16MB                        40783E3   135044.3     323360
SC31UHWM EUDSA HWM above 16MB                       20460E4   677486.5    1771920
SC24UOCC UDSA Storage Occupancy below 16MB             4666       15.5       1943
SC31UOCC EUDSA Storage Occupancy above 16MB           31378      103.9      15414
SC24SGET CDSA/SDSA GETMAINs below 16MB                     0         .0          0
SC24GSHR CDSA/SDSA storage GETMAINed below 16MB            0         .0          0
SC24FSHR CDSA/SDSA storage FREEMAINed below 16MB           0         .0          0
```

```
SC31SGET ECDSA/ESDSA GETMAINs above 16MB                    0         .0         0
SC31GSHR ECDSA/ESDSA storage GETMAINed above 16MB           0         .0         0
SC31FSHR ECDSA/ESDSA storage FREEMAINed above 16MB          0         .0         0
PCLINK   Program LINK requests                          21064       69.7      11938
PCLOAD   Program LOAD requests                          19138       63.4      11539
PCXCTL   Program XCTL requests                            152         .5          2
PCLURM   Program LINK URM requests                        303        1.0          2
PCDPL    Distributed Program Link (DPL) requests            0         .0          0
PCSTGHWM Program Storage HWM above and below 16MB       22768E4   753920.6    2084304
PC24BHWM Program Storage HWM below 16MB                235600      780.1       8680
PC31AHWM Program Storage HWM above 16MB                22746E4   753170.8    2083808
PC24CHWM Program Storage (CDSA) HWM below 16MB             0         .0          0
PC31CHWM Program Storage (ECDSA) HWM above 16MB         1900808     6294.1      17856
PC24SHWM Program Storage (SDSA) HWM below 16MB           186992      619.2        992
PC31SHWM Program Storage (ESDSA) HWM above 16MB        19229E4   636736.4    1985536
PC24RHWM Program Storage (RDSA) HWM below 16MB           52080      172.5       8680
PC31RHWM Program Storage (ERDSA) HWM above 16MB        43081E3   142653.4     157376
DB2REQCT DB2 requests                                      0         .0          0
IMSREQCT IMS (DBCTL) requests                              0         .0          0
WMQREQCT Number of WebSphere MQ requests                   0         .0          0
TCBATTCT TCBs attached count                               0         .0          0
DSTCBHWM CICS Dispatcher TCB HWM                           0         .0          0
CFCAPI   OO Foundation Class requests                      0         .0          0
SYNCPT   SYNCPOINT requests                              302        1.0          1
SOEXTRCT EXTRACT TCP/IP and CERTIFICATE requests           0         .0          0
SOCNPSCT Create Non-Persistent Outbound Socket reqs        0         .0          0
SOCPSCT  Create Persistent Outbound Socket requests        0         .0          0
SORCV    Outbound Sockets RECEIVE requests                 0         .0          0
SOSEND   Outbound Sockets SEND requests                    0         .0          0
SOTOTAL  Socket Total requests                             0         .0          0
SOCHRIN  Outbound Sockets characters received count        0         .0          0
SOCHROUT Outbound Sockets characters sent count            0         .0          0
SOMSGIN1 Inbound Sockets RECEIVE requests                  0         .0          0
```

Wait Analysis report

The Wait Analysis report shows a breakdown of dispatch and wait information for the transaction or system.

Example 7 shows the statements used to generate the report.

Example 7 Sample control statements for CICS PA Wait Analysis report

```
CICSPA IN(SMFIN001),
       APPLID(<applid>),
       LINECNT(60),
       FORMAT(':','/'),
       PRECISION(4),
       WAITANAL(OUTPUT(<ddname>),
             TITLE1('<title>'),
             SELECT(PERFORMANCE(
             INC(START(FROM(,00:00:00.00),
                         TO(,23:59:59.99)),
                  STOP(FROM(,00:00:00.00),
                         TO(,23:59:59.99))))))
```

Example 8 shows the output from a sample CICS PA Wait Analysis report, giving a breakdown of wait activity by Transaction ID.

Example 8 Sample output from the CICS PA Wait Analysis report

```
Tran=ABCD
  Summary Data                                          -------- Time ---------    ------ Count ------    ------ Ratio ------
                                                             Total     Average       Total    Average
    # Tasks                                                                            8
    Response Time                                         0.180056    0.022507
    Dispatch Time                                         0.070352    0.008794          34        4.3     39.1% of Response
    CPU Time                                              0.037625    0.004703          34        4.3     53.5% of Dispatch
    Suspend Wait Time                                     0.109703    0.013713          34        4.3     60.9% of Response
    Dispatch Wait Time                                    0.035403    0.004425          26        3.3     32.3% of Suspend
    Resource Manager Interface (RMI) elapsed time         0.000155    0.000019          48        6.0      0.1% of Response
    Resource Manager Interface (RMI) suspend time         0.000000    0.000000           0        0.0      0.0% of Suspend

  Suspend Detail                                        ------------------- Suspend Time -------------------- ----- Count -----
                                                             Total     Average   %age  Graph                    Total  Average
    JCIOWTT  Journal I/O wait time                        0.054932    0.006867  50.1% |**********                16      2.0
    RLSWAIT  RLS File I/O wait time                       0.039225    0.004903  35.8% |*******                  8      1.0
    DSPDELAY First dispatch wait time                     0.013380    0.001672  12.2% |**                       8      1.0
    TSSHWAIT Asynchronous Shared TS wait time             0.002165    0.000271   2.0% |                        2      0.3
```

Resource Usage reports

The two available Resource Usage reports show file accesses by file or by transaction.

File Usage Summary

Example 9 shows the statements used to generate the File Usage Summary.

Example 9 Sample control statements for File Usage Summary

```
CICSPA IN(SMFIN001),
APPLID(<applid>),
       LINECNT(60),
       FORMAT(':','/'),
       PRECISION(4),
       RESUSAGE(OUTPUT(<ddname>),
             FILESUMMARY(BYTRAN,TOTAL),
             TITLE1('<title>'),
             SELECT(PERFORMANCE(
             INC(START(FROM(,00:00:00.00),
                         TO(,23:59:59.99)),
                 STOP(FROM(,00:00:00.00),
                         TO(,23:59:59.99))))))
```

Example 10 shows the output from a File Usage Summary report showing file access types by transaction.

Example 10 Output from sample CICS PA File Usage Summary

```
                                  ********************** FC Calls ********************** ******** I/O Waits *******  AccMeth
Tran                #Tasks           Get      Put     Browse    Add    Delete   Total     File      RLS     CFDT   Requests
----                -------       -------- -------- -------- -------- -------- -------- -------- -------- -------- --------
ABC1                    37 Elapse Avg                                                      .0000    .0572    .0000
                           Max                                                             .0000    .2621    .0000
                       Count  Avg      236        2        2        3        0      248        0       10        0      255
                              Max      388       14       28       10        1      410        0       21        0      425


                                  ********************** FC Calls ********************** ******** I/O Waits *******  AccMeth
Tran                #Tasks           Get      Put     Browse    Add    Delete   Total     File      RLS     CFDT   Requests
----                -------       -------- -------- -------- -------- -------- -------- -------- -------- -------- --------
ABC2                  7603 Elapse Avg                                                      .0000    .0124    .0000
                           Max                                                             .0000    .4372    .0000
                       Count  Avg       51        0        6        0        0       58        0        2        0       58
                              Max      133        0     1225        0        0     1235        0       16        0     1235
```

Transaction Resource Usage List

Example 11 shows the statements used to generate the Transaction Resource Usage List. It provides a detailed list of CMF transaction resource class data. The files are reported in the sequence in which they appear in the SMF record.

Example 11 Sample control statements for Transaction Resource Usage List

```
CICSPA IN(SMFIN001),
       APPLID(<applid>),
       LINECNT(60),
       FORMAT(':','/'),
       PRECISION(4),
       RESUSAGE(OUTPUT(<ddname>),
             TRANLIST(FILE))
```

Example 12 shows output of a sample Transaction Resource Usage List. It summarizes the file activity per transaction in the sequence they appear in the SMF record.

Example 12 Sample output from CICS PA Transaction Resource Usage List

```
 V3R1M0                                                    CICS Performance Analyzer
                                                        Transaction Resource Usage List
                                                ________________________________________________
RESU0001 Printed at 13:01:46 11/16/2009    Data from 11:12:42 11/16/2009                                              Page    4188


                                Request           Fcty  Conn                                         UOW R                Response
Tran  Userid  SC TranType Term   LUName   Type   Program  T/Name Name       NETName       APPLID    Task Seq T   Stop Time    Time
---- -------- -- -------- ----  -------- ------- -------- ------ ---- ----------------- --------   ---- --- - ------------ --------
VSCA USR000A  TO UMD      <AJ0  CICSG1A1 FS:F--- LGACVS01 T/<AJ0 G1A1 GBIBMIYA.GNWS008  CICSG1D1  61344   1 T 12:22:29.122    .0167

                                ********************* FC Calls ********************* ******* I/O Waits ********  AccMeth
      File                        Get      Put     Browse     Add     Delete    Total     File      RLS      CFDT  Requests
     --------                   -------- -------- -------- -------- -------- -------- -------- -------- -------- --------
     KSDSCUST            Elapse    .0000    .0000    .0000    .0038    .0000    .0038    .0000    .0000    .0000
                         Count         0        0        0        1        0        1        0        0        0        2

     *CTLACB*            Elapse    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000
                         Count         0        0        0        0        0        0        0        0        0        1

     Total               Elapse    .0000    .0000    .0000    .0130    .0000    .0130    .0000    .0000    .0000
                         Count         0        0        0        3        0        3        0        0        0        9
```

RMF Monitor I

RMF Monitor I is a started task that writes SMF 71-79 records on a time-interval basis. These records contain overall system performance data.

RMF Monitor I can be used to report CICS transaction counts and CICS CPU usage by region. To achieve this, WLM must be configured to use reporting groups that capture resource usage for individual address spaces. To capture both CICS Transaction rates and CPU usage in the same reporting group, it is necessary to use the same reporting group name for the JES (or STC) and CICS subtype classification.

Example 13 shows an example using the TSO WLM panel to associate a report class name of CICS2A01 with a CICS region applid of IYCUZC01.

Example 13 TSO WLM panel showing report class association for a CICS region

```
          --------Qualifier--------              -------Class--------
 Action    Type       Name     Start               Service     Report
                                          DEFAULTS: CICSDEF     CICS
  ____  1  SI         IYCUZC06 ___                  CICSTRAN    ZC06
  ____  1  SI         IYCUZC25 ___                  CICSTRAN    CICS2A25
  ____  1  SI         IYCUZC10 ___                  CICSTRAN    ZC10
  ____  1  SI         IYCUZC01 ___                 CICSTRAN    CICS2A01
```

Example 14 shows an example of using the TSO WLM panel to associate a report class name of CICS2A01 with a CICS batch job called CICS2A01.

Example 14 TSO WLM panel showing report class association for CICS in JES subtype classification

```
          --------Qualifier--------              -------Class--------
 Action    Type       Name     Start               Service     Report
                                          DEFAULTS: BATCH       BATCH
  ____  1  TN         CICS2A01 ___                  CICSBTCH    CICS2A01
  ____  1  TN         CICS2A10 ___                  CICSBTCH    CICS2A10
  ____  1  TN         CICS2A25 ___                  CICSBTCH    CICS2A25
```

ERBRMFPP is the program name of the RMF post processor for SMF records. It can be used to post-process the RMF Monitor I SMF records. Example 15 provides sample statements to generate an RMF report.

Example 15 Sample control statements for ERBRMFPP RMF post processor program

```
//SYSIN    DD *
 NOSUMMARY
 SYSOUT(A)
 REPORTS(ALL)
 SYSRPTS(WLMGL(POLICY,WGPER,RCLASS,SCPER))
```

Example 16 shows example output from an RMF report using report classes to associate transaction and CPU time with an individual CICS region.

Example 16 Sample RMF report showing report class usage

```
-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
   AVG       1.00  ACTUAL                51  SSCHRT  1793  IOC      4281K  CPU  107.977  CP     28.40
   MPL       1.00  EXECUTION             18  RESP     0.2  CPU     34622K  SRB   17.835  AAPCP   0.00
   ENDED   495375  QUEUED                 0  CONN     0.1  MSO     24057M  RCT    0.000  IIPCP   0.00
   END/S  1085.80  R/S AFFIN              0  DISC     0.0  SRB      5719K  IIT    3.808
   #SWAPS       4  INELIGIBLE             0  Q+PEND   0.0  TOT     24102M  HST    0.000  AAP     0.01
   EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    52829K  AAP    0.044  IIP      N/A
   AVG ENC   0.00  STD DEV               50                               IIP      N/A
   REM ENC   0.00                                        ABSRPTN    53M
   MS ENC    0.00                                        TRX SERV   53M
```

You can see that in this run, the CICS region is executing 1085.8 transactions per second with an average response time of 51 ms, while using 28.40% of a single CP.

RMF Monitor III

RMF Monitor III can be used to monitor the performance of the coupling facility (CF). Like Monitor I, Monitor III writes SMF records that can be post-processed using ERBRMFPP, but it

also can be used interactively through TSO. With the interactive monitor, you can view real-time data for both CF performance and RLS buffer pool activity.

There are a number of useful reports that can be produced using the RMF Monitor III post-processing program ERBRMFPP. Two of these, the Structure Summary report and Cache report, are detailed in the following sections.

Example 17 shows the control statements necessary to generate the structure summary and cache reports.

Example 17 Control statements for the ERBRMFPP program to generate Structure and Cache reports

```
//SYSIN    DD *
  NOSUMMARY
  SYSRPTS(CF)
  SYSOUT(A)
```

Structure Summary report

Example 18 shows an extract of the Structure Summary report that is produced. It includes information about the two cache structures being used for RLS files and the lock structure.

Example 18 RMF Monitor III Structure Summary report output

```
STRUCTURE SUMMARY
   ------------------------------------------------------------------------------------------------------------------------
                                                  % OF            % OF   % OF     AVG    LST/DIR  DATA      LOCK     DIR REC/
         STRUCTURE                      ALLOC    CF       #      ALL    CF       REQ/   ENTRIES  ELEMENTS  ENTRIES  DIR REC
   TYPE  NAME           STATUS CHG     SIZE     STOR     REQ    REQ    UTIL     SEC    TOT/CUR  TOT/CUR   TOT/CUR  XI'S

   LOCK  IGWLOCK00      ACTIVE          4M      0.4     4393K   85.3   74.0     14546   4739        0       524K      N/A
                                                                                         131        0       674       N/A

   CACHE CACHE01        ACTIVE        245M     26.3     406074   7.9    14.0    1344.6    57K       113K     N/A        0
                                                                                          56K       112K     N/A        0
         CACHE02        ACTIVE        245M     26.3     350582   6.8    11.6    1160.9    57K       113K     N/A        0
                                                                                          56K       112K     N/A        0
```

A cache structure is a particular type of coupling facility structure that consists of one or more directory entries and optional data entries made up of one or more data elements. For more information, consult the z/OS 1.13 Information Center at this website:

http://publib.boulder.ibm.com/infocenter/zos/v1r13/index.jsp

Keep in mind that:

- When viewing a Structure Summary, look out for a high number of directory reclaims (DIR REC), which can indicate that the structure is too small.
- A high number of DIR entries compared to the number of data elements indicates that elements are being reclaimed in favor of DIR entries and that the structure is too small.

When the structure is first allocated, the ratio is defined by SMSVSAM. Over time, the ratio dynamically adjusts depending on factors such as CI size. The smaller the CI, the fewer data entries that are needed to hold the data. If the demand for storage in the structure increases, then entries are reclaimed based on a Least Recently Used (LRU) algorithm. The algorithm favors directory entries and reclaims data entries first. This is because if a data entry is reclaimed, it does not need to cross-invalidate any of the local buffers (because they are still valid). If a directory entry is reclaimed, then the control information about its related buffer is lost, which means that data integrity can be lost if all of the MVS images that had registered an interest in the buffer do not have their local copies invalidated at the same time.

Cache Structure report

Example 19 and Example 20 show report extracts for two structures. The first (Example 19) is the SMSVSAM IGWLOCK00 lock structure. The second (Example 20) is the CACHE01 user cache structure.

Example 19 Cache report for Lock structure IGWLOCK00

```
STRUCTURE NAME = IGWLOCK00          TYPE = LOCK   STATUS = ACTIVE
              # REQ    -------------- REQUESTS -------------   --------------- DELAYED REQUESTS -------------
     SYSTEM   TOTAL             #     % OF  -SERV TIME(MIC)-    REASON    #   % OF  ---- AVG TIME(MIC) -----  EXTERNAL REQUEST
     NAME     AVG/SEC           REQ    ALL     AVG   STD_DEV            REQ    REQ   /DEL    STD_DEV   /ALL   CONTENTIONS

     MV2A      4393K   SYNC   4393K    100     3.1      1.4     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL     4396K
               14546   ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED   212
                       CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0   -CONT          212
                                                                                                              -FALSE CONT    211

    ------------------------------------------------------------------------------------------------------------------------------
     TOTAL     4393K   SYNC   4393K    100     3.1      1.4     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL     4396K
               14546   ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED   212
                       CHNGD     0     0.0                      PR CMP    0    0.0    0.0        0.0    0.0   -CONT          212
                                                                                                              -FALSE CONT    211
```

Example 20 Cache report for cache structure CACHE01

```
     STRUCTURE NAME = CACHE01            TYPE = CACHE  STATUS = ACTIVE
              # REQ    -------------- REQUESTS -------------   --------------- DELAYED REQUESTS -------------
     SYSTEM   TOTAL             #     % OF  -SERV TIME(MIC)-    REASON    #   % OF  ---- AVG TIME(MIC) -----
     NAME     AVG/SEC           REQ    ALL     AVG   STD_DEV            REQ    REQ   /DEL    STD_DEV   /ALL

     MV2A       406K   SYNC    406K    100     6.1      2.2     NO SCH    0    0.0    0.0        0.0    0.0
                1345   ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0
                       CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0
                                                                DUMP      0    0.0    0.0        0.0    0.0

    ------------------------------------------------------------------------------------------------------------------------------
     TOTAL      406K   SYNC    406K    100     6.1      2.2     NO SCH    0    0.0    0.0        0.0    0.0    -- DATA ACCESS ---
                1345   ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   READS       41410
                       CHNGD     0     0.0                      PR CMP    0    0.0    0.0        0.0    0.0   WRITES     181944
                                                                DUMP      0    0.0    0.0        0.0    0.0   CASTOUTS        0
                                                                                                              XI's          500
```

The report includes the following details:

- READS represent file read requests that have not been satisfied from the buffer pool but have been read from the CF structure.
- WRITES include cases where the record was not found in the buffer pool or the CF but was found on DASD, after which it was put in the buffer pool and then written back to the CF. Application writes and rewrites are also recorded here.
- Castouts are not relevant to RLS. The RLS caching design writes *unchanged* entries to the cache. This means that the data in the cache matches the data on DASD. When unchanged entries are written, the CF can reclaim the entries, as necessary, for space (because the data is still on DASD and can be retrieved as necessary). Some XCF exploiters (such as IBM DB2®) write *changed* entries. With changed entries, the CF is not allowed to reclaim them and the exploiters must then cast out the entries to free up space. So with the RLS design, you will never see any castouts.
- XIs shown in the report represent the number of cross-invalidations that have occurred. Cross invalidation is where a CI resides in a buffer pool and another task in a different LPAR has updated a record in it. The CI is now marked as invalid and must be refreshed from the CF. XIs can also happen if the cache size is too small and directory entries get reclaimed. Buffers associated with these must be marked as XI.

RMF III TSO interface

Using TSO, if you use CLIST RMFWDM and choose option **S** for sysplex views, you get the panel shown in Example 21, which gives options for viewing the current activity in RLS.

Example 21 RMF Monitor III TSO interface Sysplex Report options

```
Sysplex Reports
         1 SYSSUM   Sysplex performance summary                (SUM)
         2 SYSRTD   Response time distribution                 (RTD)
         3 SYSWKM   Work Manager delays                        (WKM)
         4 SYSENQ   Sysplex-wide Enqueue delays                 (ES)
         5 CFOVER   Coupling Facility overview                  (CO)
         6 CFSYS    Coupling Facility systems                   (CS)
         7 CFACT    Coupling Facility activity                  (CA)
         8 CACHSUM  Cache summary                              (CAS)
         9 CACHDET  Cache detail                               (CAD)
        10 RLSSC    VSAM RLS activity by storage class         (RLS)
        11 RLSDS    VSAM RLS activity by data set              (RLD)
        12 RLSLRU   VSAM LRU overview                          (RLL)
```

This example shows the main sysplex panel from which you can choose to see details about coupling facility performance, individual cache activity, RLS activity organized by storage class and dataset, and RLS buffer pool activity.

SMSVSAM monitoring

SMSVSAM writes SMF 42 records on a time-interval basis. While data for SMF 42 records is collected by each SMSVSAM address space running within the sysplex, it is written by only one designated SMSVSAM address space.

To determine the LPAR on which the data is being recorded, issue the **D SMS,SMSVSAM,ALL** command. The SMF 42 status information shows an asterisk (*) beside the SYSNAME where recording is taking place. In Example 22, this is the LPAR named MVA0.

Example 22 Output of the D SMS,SMSVSAM,ALL command

```
DISPLAY SMS,SMSVSAM - SMF RECORD 42 STATUS
                    SMF_TIME  CF_TIME ----- SUB-TYPE  SUMMARY -----
                                          15  16  17  18  19
  SYSNAME: MVA2      YES- 4    3600- 4     YES YES YES YES YES
  SYSNAME: MVA3      YES- 3    3600- 3     YES YES YES YES YES
  SYSNAME: MVA0     *YES- 1    3600- 1     YES YES YES YES YES
  SYSNAME: MVA1      YES- 2    3600- 2     YES YES YES YES YES
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
```

The following SMS parameters affect the frequency and synchronization of SMF 42 records:

- CF_TIME: Aligns creation of all CF-related SMF 42 records with subtypes 15, 16, 17, 18, and 19, and dictates the interval at which they are written.
- SMF_TIME: Aligns SMF 42 records for DFSMS subtypes 1, 2, 15, 16, 17, 18, and 19 to the SMF_TIME interval.

Important: SMF_TIME overrides the value of CF_TIME (as well as BMFTIME and CACHETIME). To synchronize the SMF 42 records with your measurement time, you can change CF_TIME dynamically using the **SETSMS CF_TIME(*nn*)** command, where *nn* is the length of the interval in minutes.

SMSVSAM also generates SMF 42 records. Subtypes 15-19 are relevant to RLS in these ways:

- Subtype 15: VSAM RLS Storage Class Response Time Summary
- Subtype 16: VSAM RLS Dataset Response Time Summary (generation of subtype 16 records is controlled using the **V SMS,MONDS** command)
- Subtype 17: VSAM RLS Coupling Facility Lock Structure Usage
- Subtype 18: VSAM RLS CF Cache Partition Usage
- Subtype 19: VSAM RLS Local Buffer Manager LRU Statistics Summary

By default, the SMF 42 records collect data associated with storage classes. Because storage classes have numerous datasets allocated within them, this can prevent analysis of individual files. This should be considered when planning the allocation of datasets within storage classes.

Monitoring of individual datasets can be achieved using the **VSMS, MONDS(*xxx.yyy.www*),ON** command, where *xxx.yyy.www* is a dataset name.

To give an idea of the performance data available from SMSVSAM, several report extracts are presented here.

The extracts show performance data for the data component of the data set VSAMDSW2.A.CUSTOMER. The first extract shows the types and number of accesses to the buffer pool, the coupling facility, and to DASD. The second extract shows information about the locks for the data component. The third extract shows information about the buffer pool.

These three extracts are also generated for the index component of the dataset, but these particular index component extracts are not included in this paper.

It should be noted that in the full report, the data and index extracts are repeated for the sysplex view and for all LPAR views.

Sysplex Wide Data Set summary

The first extract (Example 23) is based at the data set level from SMF 42 subtype 16 records. It shows the types and number of accesses to the buffer pool, the coupling facility, and to DASD.

Example 23 Sysplex Wide Data Set summary

```
*** SYSPLEX WIDE DATA SET SUMMARY         (BELOW BAR) ***

   (SMF42GAA) INTERVAL DURATION (SECONDS).........:           300
   (SMF42GAB) DATA SET Name.......................: VSAMDSW2.A.CUSTOMER.DATA
   (SMF42GAC) VSAM Sphere name....................: VSAMDSW2.A.CUSTOMER
   (SMF42GAE) Storage class name..................: SXPXXSO2
   (SMF42GAF) Cacheset name.......................: CS2
   (SMF42GAH) DFP Cache Structure name............: CACHEO2    COMPONENT-DATA
   (SMF42GAJ) SMS > 4K CF caching status..........: GT4KNOTACT

   *** NORMAL DIRECT ACCESS SUMMARY ***
   (SMF42GCB) total number of requests............:        17,413
   (SMF42GCC) total read requests (NRI)...........:         4,797
   (SMF42GCD) total read requests (CONSISTENT)....:         7,998
   (SMF42GCE) total write requests................:         4,618
```

```
   (SMF42GCF) number of BMF requests..............:        17,528
   (SMF42GCG) number of BMF read requests.........:        12,856
   (SMF42GCH) number of BMF write requests........:         4,672
   (SMF42GCI) number of BMF read hits.............:        12,799
   (SMF42GCJ) number of BMF valid read hits.......:        12,796
   (SMF42GCK) number of BMF false invalids........:             0
   (SMF42GCL) # reqs processed by Plex Cache Mgr..:         4,718
   (SMF42GCM) number of CF  read requests.........:            60
   (SMF42GCN) number of CF  write requests........:         4,658
   (SMF42GCO) number of CF  read hits.............:            55
   (SMF42GCP) number of CF  read castins..........:             5
   (SMF42GCQ) Bytes xferred to CF cache structure.:        20,480
   (SMF42GCR) Number of read  real I/O to DASD....:             5
   (SMF42GCS) Number of write real I/O to DASD....:         4,658
   (SMF42GCT) Bytes xferred to DASD for READ......:        20,480
   (SMF42GCU) Bytes xferred to DASD for WRITE.....:    19,079,168
   (SMF42GCW) Time for all requests in INTVL (mS).:         5,877
   (SMF42GCX) Avg. response time (TIME/REQS)......:             0
   (SMF42GCY) Norm response time (TIME/BYTES/4K)..:             1
```

Lock Statistics summary

The second extract (Example 24) shows information about the locks for the data component.

Example 24 Lock Statistics summary

```
*** LOCK STATISTICS SUMMARY ***
   (SMF42GPA) # of record lock reqs (OBT/ALT/PROM):         8,018
   (SMF42GPB) # reqs with true lock contention....:             0
   (SMF42GPC) # reqs with false lock contention...:             0
   (SMF42GPD) # of record lock release requests...:         7,962
   (SMF42GPE) # of component_1 locks requests.....:            29
   (SMF42GPF) # of component_1 release requests...:             0
   (SMF42GPH) # of comp_1, class_1, DIWA locks....:            29
   (SMF42GPI) # DIWA locks - true contention......:             0
   (SMF42GPJ) # DIWA locks - false contention.....:             0
   (SMF42GPK) # DIWA locks - release locks........:             0
   (SMF42GPL) # of comp_1, class_2, UPGRADE locks.:             0
   (SMF42GPM) # UPGRADE locks - true contention...:             0
   (SMF42GPN) # UPGRADE locks - false contention..:             0
   (SMF42GPO) # UPGRADE locks - release locks.....:             0
   (SMF42GPP) # of comp_1, class_3,PREFORMAT locks:             0
   (SMF42GPQ) # PREFORMAT locks - true contention.:             0
   (SMF42GPR) # PREFORMAT locks - false contention:             0
   (SMF42GPS) # PREFORMAT locks - release locks...:             0
   (SMF42GPT) # of component_2 locks requests.....:             0
   (SMF42GPU) # Comp_2 locks - true contention....:             0
   (SMF42GPV) # Comp_2 locks - false contention...:             0
   (SMF42GPW) # Comp_2 locks - release locks......:             0
```

This extract illustrates several important aspects of locking:

- Locking of individual records within VSAM data sets is a central element of the overall design. RLS also uses other, higher level locks for serializing operations such as control interval and control area split.

- DIWA Locks (Data Insert Work Area) are for Control Interval (CI) splits and Entry Sequenced Data Set (ESDS) Add to End.
- PREFORMAT is when a new CA has been acquired.
- UPGRADE is for IBM AIX® support.

Buffer Management Facility (BMF) LRU statistics

The third extract (Example 25) shows information about the buffer pool. The SMF record number is also shown in the report and can be used to reference the description.

Example 25 Dataset BMF LRU statistics

```
*** DATASET BMF LRU STATISTICS ***
   (SMF42GRA) Number of REDOs.....................:            14
   (SMF42GRB) Number of Recursive REDOs...........:             0
   (SMF42GRC) Number of BMF writes................:         4,619
   (SMF42GRD) Number of SCM reads.................:            60
   (SMF42GRE) Number of SCM reads (castout lock)..:             7
   (SMF42GRG) Percentage REDOs....................:             0
   (SMF42GRH) Percentage Recursive REDOs..........:             0
   (SMF42GRI) Percentage SCM CASTOUT LOCK.........:             3
```

This extract illustrates several important aspects of BMF LRU statistics:

- RLS maintains buffer coherency across the local buffer pools by using the buffer registration and invalidation functions of the CF cache.
- There are multiple copies of a CI. Transactions have their own copy of the CI and each might be updating a different record within it.
- The castout lock is held while a CI is written to DASD by the first user to rewrite the CI. This prevents multiple users writing changes to the same CI at the same time. The subsequent attempts fail when they attempt to write and the copy they have is merged with the new one. This is called a *redo* and is handled in the background, without the involvement of the application program.
- The redo is retired every 1.5 ms while the castout lock is held. It is possible, therefore, to get recursive redos.
- In Example 25, the reference to SCM reads (castout lock) represents the number of times someone tried to read a CI but the castout lock was held.

Additional information

The following sections provide additional information you might find useful in setting up and managing CICS Transaction Server and its related products.

Useful commands

- **D SMS,SMSVSAM,ALL**: Show prime SMSVSAM server in sysplex (the SMF writer).
- **D SMS,CFCACHE(*)**: Show status of RLS CF cache structures.
- **D SMS,CFLS**: Show lock rates and contention on lock structure.
- **V SMS,MONDS('dsname'),ON**: Start dataset monitoring.
- **D SMS,MONDS(*)**: Show which datasets are being monitored.

- **SETSMS RLS_MAX_POOL_SIZE(nn)**: Change the size of the RLS buffer pool.
- **SETSMS RLSABOVETHEBARMAXPOOLSIZE(sysid,nn)**: Change the size of the RLS buffer pool above the bar.
- **SETSMS RLSFIXEDPOOLSIZE(sysid,nn)**: Allocate fixed real storage to the buffer pool.
- **SETSMS CF_TIME(600)**: Change the interval for recording SMSVSAM performance data.
- **V SMS,CFCACHE(CACHE02),ENABLE**: Enable a CF cache for RLS use.
- **D SMS,OPTIONS**: Show all the current SMSVSAM options.
- **D SMF,O**: Show the current SMF settings.
- **SETXCF START,ALTER,STRNM='strname',SIZE=nnn**: Change the size of a structure dynamically.
- **ISPF D.1.5**: Display characteristics of a storage class, including the nominated cacheset.
- **ISPF 8.6**: Display the CF structures associated with a cacheset.

RLS terminology

Record lock	An XES lock resource obtained by SMSVSAM on behalf of a user and associated with a logical record. The lock resource name is based on a 16-byte hashed version of the record's key (or RBA, or RRN), as well as the sphere name and component name. There are also other locks to serialize splits, for example.
True contention	When two different users or processes attempt to access the same record lock at the same time. Reported as *true contention* on the **D SMS,CFLS** command and in RMF reports. If you have high true contention rates, you need to tune your application or use NRI for reads.
False contention	When VSAM RLS assigns locked resources to an entry value in the lock table, and uses this entry value to quickly check whether a resource is already locked. If the lock structure (and thus the lock table) is too small, many locks can be represented by a single value, making *false* lock contention possible. False lock contention occurs when two different locks on different resources attempt to use the same lock entry. The second lock requester is suspended until VSAM RLS determines that there is no real lock contention on the resource. This is reported as false contention on the **D SMS,CFLS** command. For high false contention rates, you need to create a larger lock structure. A True/False or False/False contention occurs when either true contention or false contention occurs, and the holder releases the record before the RLS contention exit runs. At this point, RLS cannot differentiate between true or false contentions. This type of contention is reported as false contention on the **D SMS,CFLS** command.
Deadlocks	When two different transactions, each holding a record lock, attempt to obtain the other transaction's record lock. SMSVSAM abnormally terminates one of the waiting lock requests (RPL RC=8 RSN=21), based on the Deadlock_detection value in your SYS1.PARMLIB IGDSMSxx member.
Retained lock	A record lock obtained by a recoverable subsystem, such as CICS, and which was still held at the time of a failure in CICS, IGWLOCK00, CF, z/OS, or other SMSVSAM or Indoubt transactions. A job can be

cancelled without retained locks because this job does not register as subsystem with SMSVSAM.

Lost Locks — A sphere is said to be in a *lost locks* state if the sphere was being accessed by a recoverable subsystem when a failure in the lock structure occurred at the same time as a failure in at least one of the SMSVSAM address spaces (in other words, a double failure scenario). Use the **`IDCAMS SHCDS LISTSUBSYS(ALL)`** command to list CICS subsystems holding retained or lost locks. This situation can be avoided by duplexing or failure isolation. CEMT INQ DSNAME can also give additional information about data sets in lost lock status.

Reference documentation

The following publications provide additional detail on the topics covered in this paper:

- *VSAM Demystified*, SG24-6105

  `http://www.redbooks.ibm.com/abstracts/sg246105.html`
- *CICS and VSAM Record Level Sharing: Recovery Considerations*, SG24-4768

  `http://www.redbooks.ibm.com/abstracts/sg244768.html`
- *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766

  `http://www.redbooks.ibm.com/abstracts/sg244766.html`
- *CICS and VSAM Record Level Sharing: Planning Guide*, SG24-4765

  `http://www.redbooks.ibm.com/abstracts/sg244765.html`

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

John Burgess is a senior software engineer working in the Hursley CICS performance team. He has over 20 years of experience with CICS and specializes in performance of large systems.

Arndt Eade is a senior software engineer working in the CICS services and technology team at IBM Hursley laboratory in the United Kingdom. He has over 25 years of IT experience, mainly on IBM zSeries®, working on products such as CICS and IBM WebSphere® MQ. He joined IBM in 2003.

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include 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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4905-00 was created or updated on December 20, 2012.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks