Redpaper

John Burgess
Trevor Clarke
Arndt Eade

IBM CICS Performance Series: CICS, DB2, and Thread Safety

This IBM® Redpaper™ publication highlights the findings of a study about the factors that affect the performance of IBM CICS® transactions that access IBM DB2® resources through the CICS DB2 attachment facility. Most of the study used the following hardware and software:

- A 2097-E64 logical partition (LPAR) with three dedicated central processors

  For Large Systems Performance Reference (LSPR), 2097-703 was used.
- IBM z/OS® 1.11
- IBM CICS Transaction Server V4.1
- IBM DB2 V9

CICS Transaction Server V4.2: Since the writing of this paper, CICS Transaction Server V4.2 was announced. For more information, see the CICS Transaction Server for z/OS, Version 4 Release 2 at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

This paper can assist application designers and developers in developing CICS applications that perform optimally within the CICS environment. It can also provide guidance to CICS systems programmers and administrators about how to configure regions to optimize access to DB2 subsystems.

This paper is one of a series focused on CICS performance written by members of the IBM Hursley CICS development community. The subject matter in this series is based on feedback from CICS customers.

Additional information: To gain a thorough understanding of the CICS DB2 attachment facility and to avoid unnecessary repetition of information, see the *CICS Transaction Server for z/OS V4.2: DB2 Guide*, SC34-7164, at:

http://www.ibm.com/support/docview.wss?uid=pub1sc34716400

Strategy for defining and configuring the connection

When defining, configuring, and tuning CICS DB2 connections, approach the task holistically, and involve both the CICS *and* the DB2 systems programming teams. In this context, *holistic* refers to the LPAR or SYSPLEX level. It involves understanding the affect that a change can have on the system as a whole, whether introducing a new application or creating a CICS DB2 infrastructure.

Generally, new CICS transactions that access DB2 are defined to use pool threads from the DB2 connection pool, and packages are bound with **ACQUIRE(USE)** and **RELEASE(COMMIT)**. From the CICS perspective, this design means that only a DB2CONN definition is required. From a concurrency perspective, the **THREADLIMIT** and **TCBLIMIT** parameters (and the MAXTASK and MAXOPENTCBS values) control the maximum number of threads that can be active at any one time. In this environment, apply protected entry threads, **RELEASE(DEALLOCATE)**, and DB2 package BIND options selectively.

The collection and analysis of CICS and DB2 statistics and monitoring data is key to understanding and successfully configuring the CICS and DB2 infrastructure. Products, such as CICS Performance Analyzer 3.2 and IBM OMEGAMON® XE for DB2 Performance Expert on z/OS 5.1, provide reports that assist with the analysis of this data.

In addition, during testing, or as a result of performance observations in the production environment, it might be beneficial to define DB2ENTRY and DB2TRAN definitions for a particular transaction or set of transactions.

Figure 1, taken from *CICS Transaction Server for z/OS V4.2: DB2 Guide*, SC34-7164, provides initial guidance on using protected threads and includes suggestions for the correct bind options to use.

Transaction Description	**Thread type**	**Overflow**	**Acquire**	**Release**
High volume (all types)	Protected entry	Note 1	ALLOCATE	DEALLOCATE
Terminal-oriented with many commits (plus terminal if NONTERMINAL=YES)	Protected entry	Note 2	Note 3	DEALLOCATE
Low volume, requires fast response time	unprotected entry	Yes	USE	COMMIT
Low volume, limited concurrency	unprotected entry	Never	USE	COMMIT
Low volume, does not require fast response time	Pool	Not applicable	USE	COMMIT
Non-terminal-oriented with many commits (NONTERMREL=NO)	Note 4	Note 4	Note 3	DEALLOCATE

Notes:

1. Yes, but define enough entry threads so this happens infrequently
2. Yes, but if it overflows to the pool no protected thread is used.
3. ALLOCATE if most of the SQL in the plan is used. Otherwise use ACQUIRE(USE).
4. Threads are held until EOT. Use pool threads for a short transaction. Consider entry threads for longer running transactions.

Figure 1 CICS DB2 thread and bind suggestions

When considering a strategy for the connection, keep in mind the following points:

- The use of the BIND option **`ACQUIRE(ALLOCATE)`** applies to *plans* and not packages. The option is not supported in DB2 V10.
- Independent of **`RELEASE(COMMIT|DEALLOCATE)`**, apply thread reuse to high volume transactions. This case applies to transactions per second (TPS) rates in excess of 10 and not for transactions with TPS rates less than 1, because protected threads are a precious resource.
- **`RELEASE(COMMIT|DEALLOCATE)`** is of consequence only if thread reuse is being achieved. When thread reuse is achieved, the respective transaction must apply **`RELEASE(DEALLOCATE)`** only to frequently used packages.
- Thread reuse can be achieved without the use of protected threads by queuing requests to the pool and entry thread. However, this use does require careful planning and monitoring.

CICS and DB2 configurations

Figure 2 shows the resources that are required by CICS to connect to DB2.

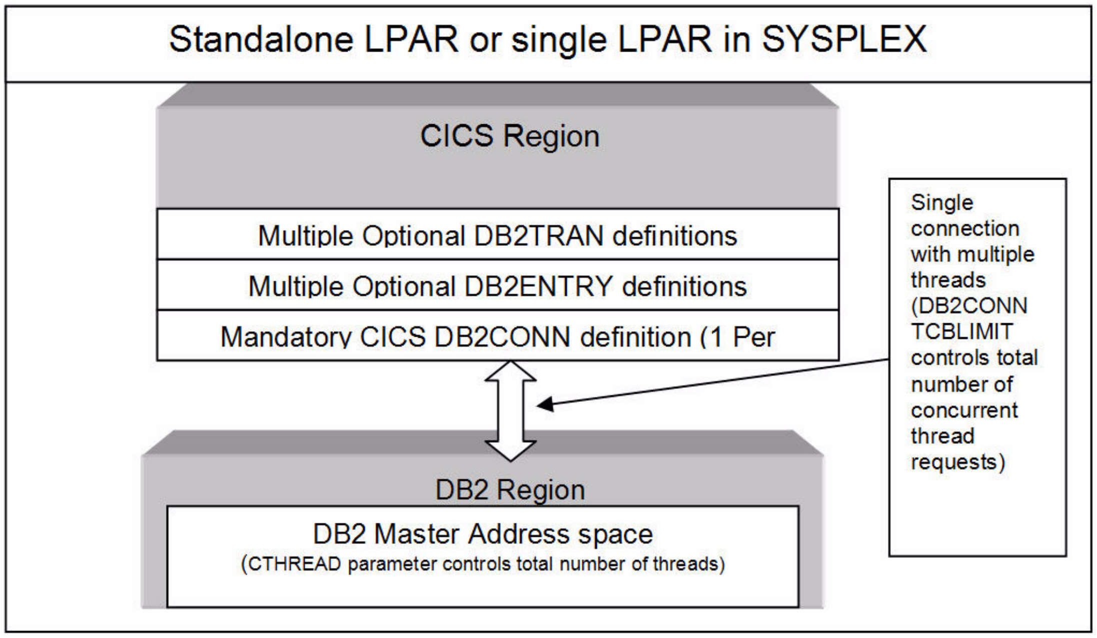

Standalone LPAR or single LPAR in SYSPLEX
CICS Region
Multiple Optional DB2TRAN definitions
Multiple Optional DB2ENTRY definitions
Mandatory CICS DB2CONN definition (1 Per
Single connection with multiple threads (DB2CONN TCBLIMIT controls total number of concurrent thread requests)
DB2 Region
DB2 Master Address space
(CTHREAD parameter controls total number of threads)

Figure 2 Resources used by CICS when connecting to DB2

Figure 3 shows a typical configuration where multiple CICS regions attach to a single DB2 instance.

CTHREAD value: The `CTHREAD` value defines the total number of allied threads. It does not include distributed (DDF) threads.

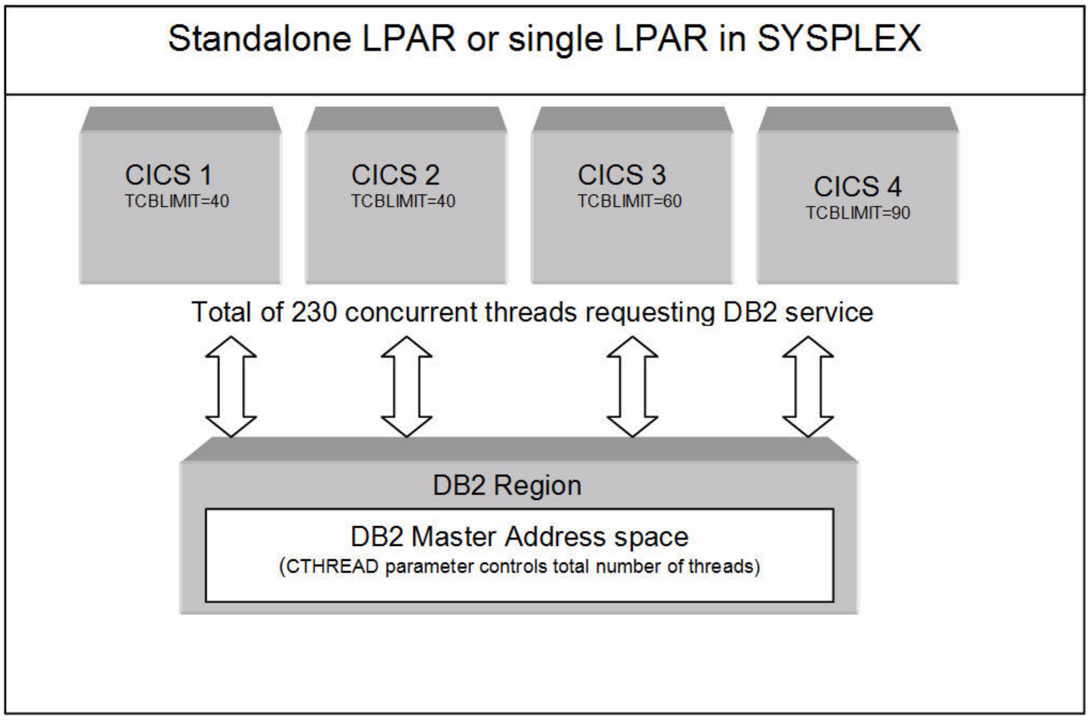

Standalone LPAR or single LPAR in SYSPLEX
CICS 1
TCBLIMIT=40
CICS 2
TCBLIMIT=40
CICS 3
TCBLIMIT=60
CICS 4
TCBLIMIT=90
Total of 230 concurrent threads requesting DB2 service
DB2 Region
DB2 Master Address space
(CTHREAD parameter controls total number of threads)

Figure 3 Multiple CICS regions connected to a single DB2 instance

Figure 4 illustrates a more typical SYSPLEX environment, where CICS regions and DB2 subsystems run within multiple LPARs. Availability requirements usually result in cloned CICS regions that are connected to one or more DB2 subsystems that are part of a DB2 data sharing group.

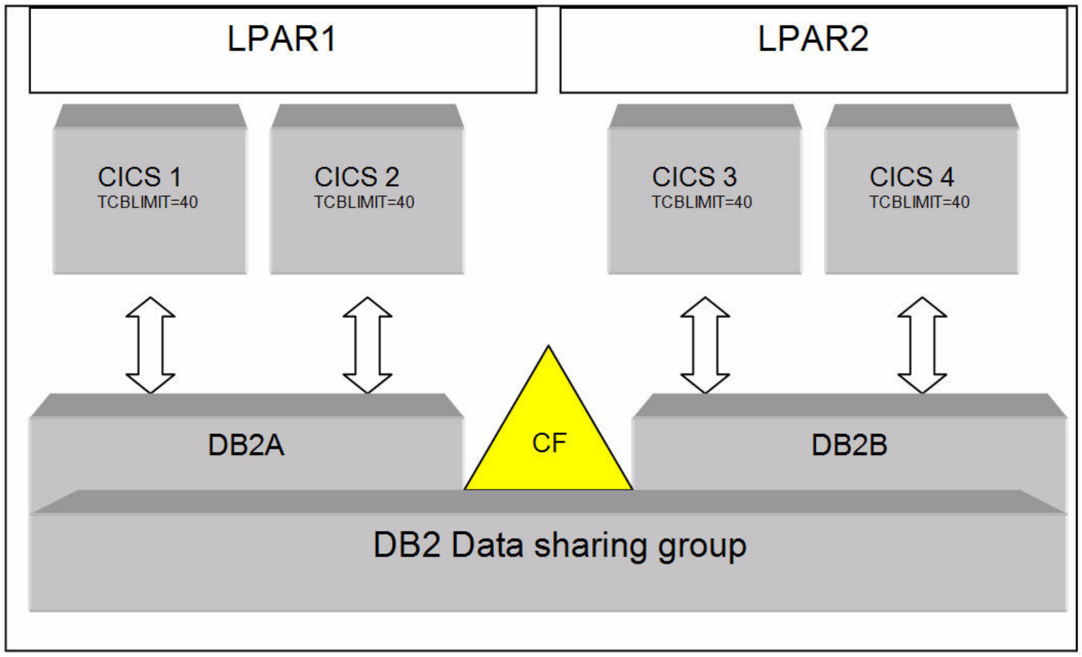


Figure 4 Multiple CICS regions connected to DB2 data sharing group in SYSPLEX environment

When administering such environments, activity in any of the CICS regions, or against any of the DB2 subsystems to which they are connected, such as batch jobs or utilities, can affect the performance across all regions. Consider a balanced design for CICS application-owning regions (AORs) that access a single DB2 subsystem.

Before DB2 V10, the practical limit on DB2 threads was determined by the average thread footprint and the 31-bit virtual storage that was available to the DBM1 address space. In DB2 V10, the virtual storage constraint is removed with the potential constraint now on the real storage that is available to the LPAR.

Performance measurements

Several performance benchmarks were performed to support the information provided in this paper. The workload for this paper is referred to as the *RTW workload*. This workload is one of the workloads that is used by the CICS performance team for release-to-release performance measurement. The RTW workload consists of the following components:

- Seven separate transactions, each of which starts two threadsafe programs
- For each transaction, a COBOL program that has minimal business logic and that links to a second COBOL program
- A second COBOL program, which issues approximately 200 DB2 calls
- DB2 SQL calls that are a mixture of OPEN, FETCH, CLOSE, INSERT, UPDATE, and DELETE calls
- Five of the seven transactions that run as a single unit of work (UOW), with the remaining two transactions that span multiple UOWs

CICS and the DB2 attachment facility

To access DB2 subsystems from a CICS address space, the CICS-DB2 attachment facility must be installed and connected to a DB2 subsystem. Remember that a CICS transaction that accesses DB2 requires a DB2 thread. In addition, the CICS DB2 attachment facility uses an open transaction environment (OTE). Therefore, all requests that enter the DB2 task-related user exit (TRUE) occur on an L8 task control block (TCB), irrespective of the API, EXECKEY, and CONCURRENCY program attributes or the language of the program that is making the call.

Additional information: For an understanding of when threads are allocated, released, reused, or deallocated, see the *CICS Transaction Server for z/OS V4.2: DB2 Guide*, SC34-7164.

The DB2CONN definition contains parameters that control the maximum number of threads and TCBs that can be used to access DB2. These parameters, with the **MAXTASK** and **MAXOPENTCBS** parameters, ultimately control the throughput and concurrency of requests from a single CICS region.

With installations that implement configurations such as the configurations in Figure 3 on page 4 and Figure 4 on page 5, the total number of concurrent threads is the sum of all potential threads for all connected CICS regions. In a SYSPLEX, where multiple DB2s might be available, the total concurrent requests that access a single DB2 can be the sum of all connections across all CICS regions that are hosted on multiple LPARs. For cloned CICS regions, keep in mind these parameters when changing shared DB2CONN definitions or when using DB2 group attach.

Irrespective of the number of threads and TCBs that are defined to an individual or group of CICS regions, the maximum concurrently allocated threads that are allowed by a DB2 subsystem is controlled by the DB2 **CTHREAD** subsystem parameter.

Important: Setting unrealistic thread limits is a common cause of sporadic response times, poor CICS and DB2 performance, and outages in CICS and DB2 availability.

Group attach facility

The group attach facility is provided by DB2 and is used by CICS. By using the group attach facility, a DB2 data sharing group name rather than a specific DB2 subsystem name can be specified on the DB2CONN definition. Thus, the CICS system administrator does not need to hardcode a specific DB2 subsystem name. When specified, code in DB2 determines whether an eligible DB2 subsystem is available within the data sharing group and establishes a connection. This setting is useful in a highly available environment where multiple DB2 subsystems can be hosted on single or multiple LPARs. Be aware, however, of the recovery implications of using the group attach facility and, as mentioned previously, potential concurrency problems.

Consider the typical CICS configuration shown in Figure 5.

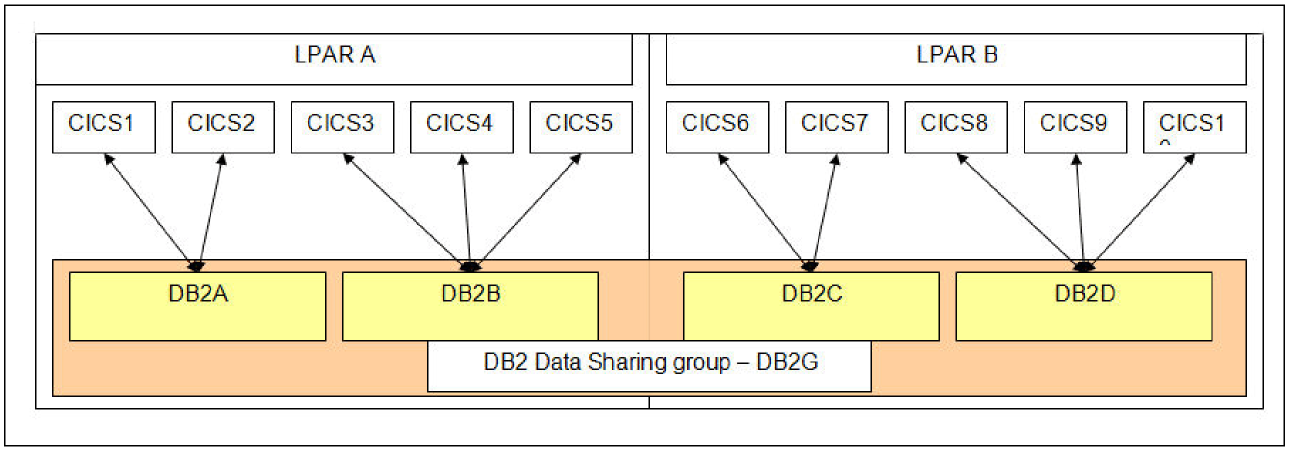


Figure 5 Typical CICS configuration in a SYSPLEX environment

In this example, all CICS regions (CICS1 to CICS10) specify a group attach name (DB2G) on their DB2CONN definitions, rather than individual DB2 subsystem names (such as DB2A, DB2B, DB2C, or DB2D). In general, you might expect an even distribution of connections, as shown in the diagram. If a DB2 subsystem failure occurs, such as the loss of DB2A, problems can arise when CICS1 and CICS2 regions attempt to attach again. In this example, a connection is established to DB2B, which if not correctly configured, might be unable to cope with the increase in threads and workload. This increase, in turn, can cause a second DB2 outage, which leaves all CICS regions on LPAR A in a position where no eligible DB2 subsystem is available.

Use fixed CICS-DB2 connections: Ensure that the DB2 subsystems, to which CICS regions connect, can support the maximum connections, both in normal operating and failover scenarios. For this reason, do not use a DB2 group attach instead of fixed CICS-DB2 connections.

Subgroup attach support in DB2 V10: In DB2 V10, more organization and control over connection requests are provided by the implementation of SubGroup attach support. For more information, see the DB2 V10 for z/OS Information Center at:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc/src/alltoc/db2z_10_prodhome.htm

CICS support for this feature was provided in CICS Transaction Server V4.2 and retrofitted to CICS Transaction Server V4.1 and CICS Transaction Server V3.2 through the service channel. The following APARs are required:

- For CICS Transaction Server 4.1, APAR PM25602/PTF UK62289
- For CICS Transaction Server 3.2, APAR PM25688/PTF UK62398

Key attributes that affect performance

The settings of the DB2CONN and DB2ENTRY attributes can significantly affect the performance of CICS transactions, as described in the following sections.

DB2CONN

The following DB2CONN settings can affect the performance of CICS transactions:

- **`TCBLIMIT`**
  - This setting indicates the maximum number of L8 TCBs that can concurrently establish a connection with DB2. If the number of transactions that simultaneously requests DB2 services exceeds this number, the excess transactions are queued. The CICS wait state that is associated with this queue is DB2CONWT.
  - The sum of all pool and entry threads must not exceed this value.
- **`THREADLIMIT`**
  - This setting controls the maximum number of threads that are defined to the pool (DB2CONN) or entry (DB2ENTRY).
  - Requests to the pool in excess of the **`THREADLIMIT`** can be queued or abended.

> **THREADLIMIT value:** CICS does not allow the **`THREADLIMIT`** value to exceed the **`TCBLIMIT`** value of the DB2CONN definition. However, you can define additional DB2ENTRY threads that cause the total **`THREADLIMIT`** to exceed the **`TCBLIMIT`**. In this case, no more than **`TCBLIMIT`** threads are allowed.
>
> For example, a DB2CONN is initially defined that specifies **`TCBLIMIT=500`**, **`THREADLIMIT=500`**. After some performance analysis, two DB2ENTRY definitions are created that each specify **`THREADLIMT=100`**. The total number of threads defined is now 700, but the **`TCBLIMIT`** remains at 500.

- **`PRIORITY`**
  - This setting determines the priority of the L8 TCB in relation to the QR TCB.
  - Upon startup of the CICS-DB2 attachment facility, the QR TCB priority is reduced by 3. An L8 TCB that accesses DB2 running with thread **`PRIORITY(HIGH)`** has a dispatch priority of 2 above the QR TCB. An L8 TCB that accesses DB2 running with thread **`PRIORITY(LOW)`** has a dispatch priority of 2 below the QR TCB.

> **PRIORITY setting:** After the PRIORITY setting is assigned, it remains active for the duration of the transaction, even when running user code.

Some debate surrounds the correct setting for the **`PRIORITY`** attribute. From a DB2 perspective, use the default of HIGH. However, from a CICS perspective, the use of a **`PRIORITY(HIGH)`** setting, especially in systems with high CPU utilization, can lead to contention for resource between the QR and L8 TCBs. The occurrence of this condition can lead to low dispatch to CPU ratios on the QR TCB.

To avoid starving the QR TCB of CPU resource, do not run all DB2 threads with **`PRIORITY(HIGH)`**. For more information about the use of this attribute, see *CICS Transaction Server for z/OS: CICS Performance Guide*, SC34-6452.

Also, keep in mind the following CICS Transaction Server V4.2 enhancements:

- The minimum specifiable PURGECYCLE value is reduced from 30 seconds to 5 seconds. The default remains at 30 seconds.
- A new REUSELIMIT attribute controls the number of times an unprotected or protected thread can be reused before it is terminated. The range 0-10000 can be specified. A value of 0 indicates no limit, which is the situation before CICS Transaction Server V4.2.

  Default value: The default value for the REUSELIMIT attribute is 1000.

  Important: Give careful consideration to the setting of the REUSELIMIT attribute. For information about the correct setting of this attribute, see the *CICS Transaction Server for z/OS V4.2: Resource Definition Guide*, SC34-7181, at:

  `http://www.ibm.com/support/docview.wss?uid=pub1sc34718100`

DB2ENTRY

The following settings for DB2ENTRY can affect the performance of CICS transactions:

- `PROTECTNUM`

  This attribute specifies the number of threads that are protected for the DB2ENTRY. A thread, when released by a transaction and no other work is queued, can be protected, meaning that it is not terminated immediately. Rather, it remains in a dormant state, ready for reuse, until two complete purge cycles (specified by the PURGECYCLE attribute on the DB2CONN definition) have elapsed.

- `THREADWAIT`

  When all entry threads are in use, this setting indicates to take one of the following actions for a new thread request:

Queue the request	`THREADWAIT(YES)`
Abend the request	`THREADWAIT(NO)`
Overflow the request to the pool	`THREADWAIT(POOL)`

  When threads are overflowed to the pool, the plan, accounting, and authorization settings from the DBENTRY attribute are carried with it. Pool thread reuse can be affected if the plan name differs from the name that is assigned to the pool thread, because threads can be reused only if the plan name remains unchanged.

Dynamic plan exit

The use of dynamic plan exits was an interim solution to address issues in the CICS DB2 environment before packages were available in DB2. Thus, use packages instead of the DB2 plan exit. In some installations, the dynamic plan exit is used as a mechanism to assign the schema name for applications that use unqualified object names.

The *CICS Transaction Server for z/OS V4.2: DB2 Guide*, SC34-7164, contains guidance about the use of the `SET CURRENT PACKAGESET` parameter to assign the collection name that is used by an application. This technique removes the need for the dynamic plan exit but requires an application change.

Frequently changing the plan name reduces the likelihood of thread reuse. Therefore, a single static plan is ideal.

Sign-on, partial sign-on, and ACCOUNTREC

The value specified on the SIGNON attribute of the DB2CONN and DB2ENTRY definitions can significantly affect the amount of security checking that takes place.

A (full) sign-on is performed by DB2 when the AUTHID associated with the DB2 request changes, irrespective of whether the thread is created or reused. The more frequently the AUTHID changes, the greater the number of sign-on requests that occur.

In comparison, a partial sign-on is performed to cause DB2 accounting records to be created. By default, DB2 creates accounting records at thread termination and when the authorization ID changes. Additional accounting records, driven by partial sign-on, are generated depending upon the **`ACCOUNTREC`** setting on the DB2CONN and DB2ENTRY definitions.

The default **`ACCOUNTREC`** setting of `NONE` results in no additional DB2 accounting records being generated. An **`ACCOUNTREC`** setting of `UOW` or `TASK` results in the generation of at least one DB2 accounting record per CICS transaction.

When using the **`ACCOUNTREC`** setting, keep in mind the following considerations:

- In the CICS statistics report, the partial sign-on count is a subset of the (full) sign-on count.
- The use of the **`ACCOUNTREC(NONE)`** setting significantly reduces the value of DB2 accounting. To allow DB2 performance monitoring and problem determination, use the **`ACCOUNTREC(UOW)`** or **`ACCOUNTREC(TASK)`** settings.

CICS DB2 applications

DB2 requests made from a CICS transaction, irrespective of the API, EXECKEY, and CONCURRENCY attribute settings, are switched to run on an L8 TCB in CICSKEY when entering the DB2 TRUE. Depending on the program attributes and the other types of calls made by the application program before and after the request, up to two TCB switches might be required to achieve this request. The cost of a single TCB switch is approximately 2,000 instructions, which can lead to a possible cost of approximately 4,000 instructions if two switches are required.

To take advantage of the benefits of storage protection, transaction isolation, and OTE, define application programs that are running in CICS by using the **`API(CICSAPI)`**, **`EXECKEY(USER)`**, and **`CONCURRENCY(THREADSAFE)`** settings. The **`CONCURRENCY(THREADSAFE)`** setting depends on whether the program is coded to threadsafe standards. With these settings, CICS normally starts the initial program under the QR TCB but switches to the L8 TCB as required.

When running on the L8 TCB, CICS continues to run the application on the L8 TCB until it is necessary to switch back, for example to run a non-threadsafe API request. Observations of customer data show that the number of switches incurred, and the cost incurred by the switch, can be significant. The number of switches is seen in the SMF 110 monitoring data as a high DSCHMDLY count.

Figure 6 shows the TCB switching that typically occurs for a nonthreadsafe CICS DB2 application. Notice that all **EXEC CICS** commands are run by using the QR TCB with task switches to an L8 TCB that occurs for SQL requests to DB2.

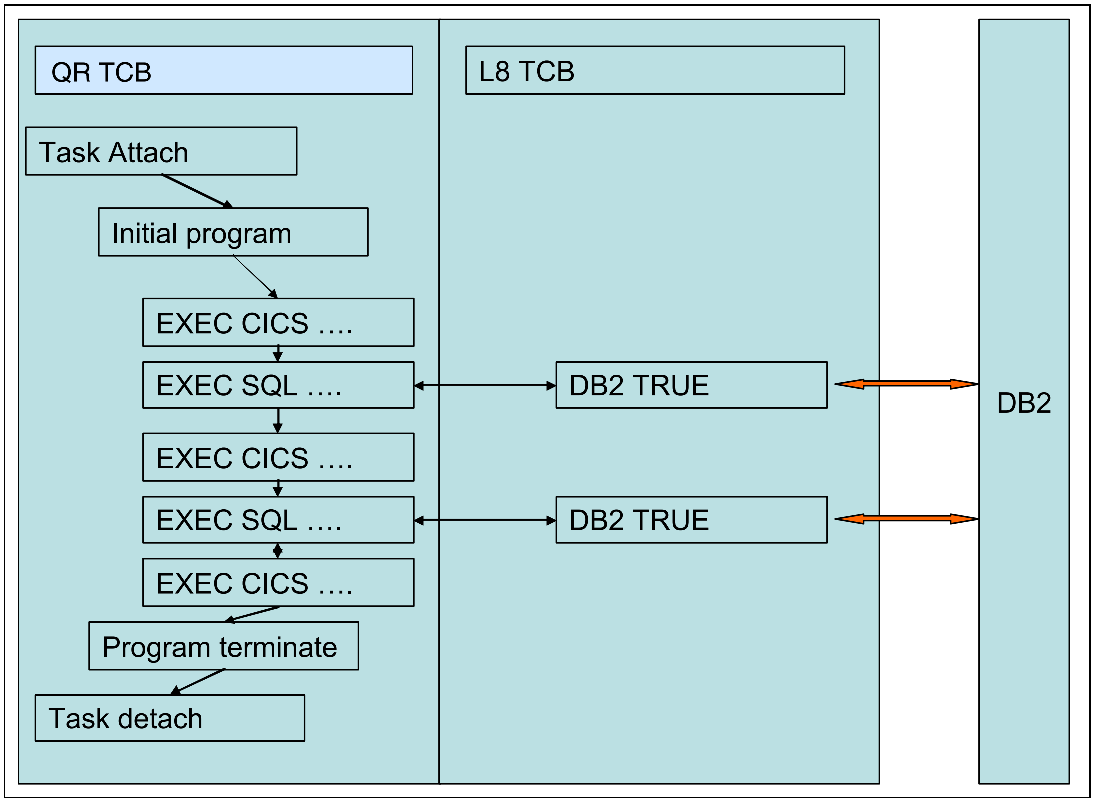


Figure 6 Typical TCB usage for a nonthreadsafe CICS DB2 transaction

Compare Figure 6 on page 11 with Figure 7 where the program is defined as a threadsafe transaction. Notice that, similar to the nonthreadsafe transaction, task attach and initial program execution occur on the QR TCB. When DB2 access is required, the transaction is switched to an L8 TCB, where it remains until it is necessary to switch back to the QR. In this example, the transaction needs to switch back to the QR for the synchronization point and task termination processing.

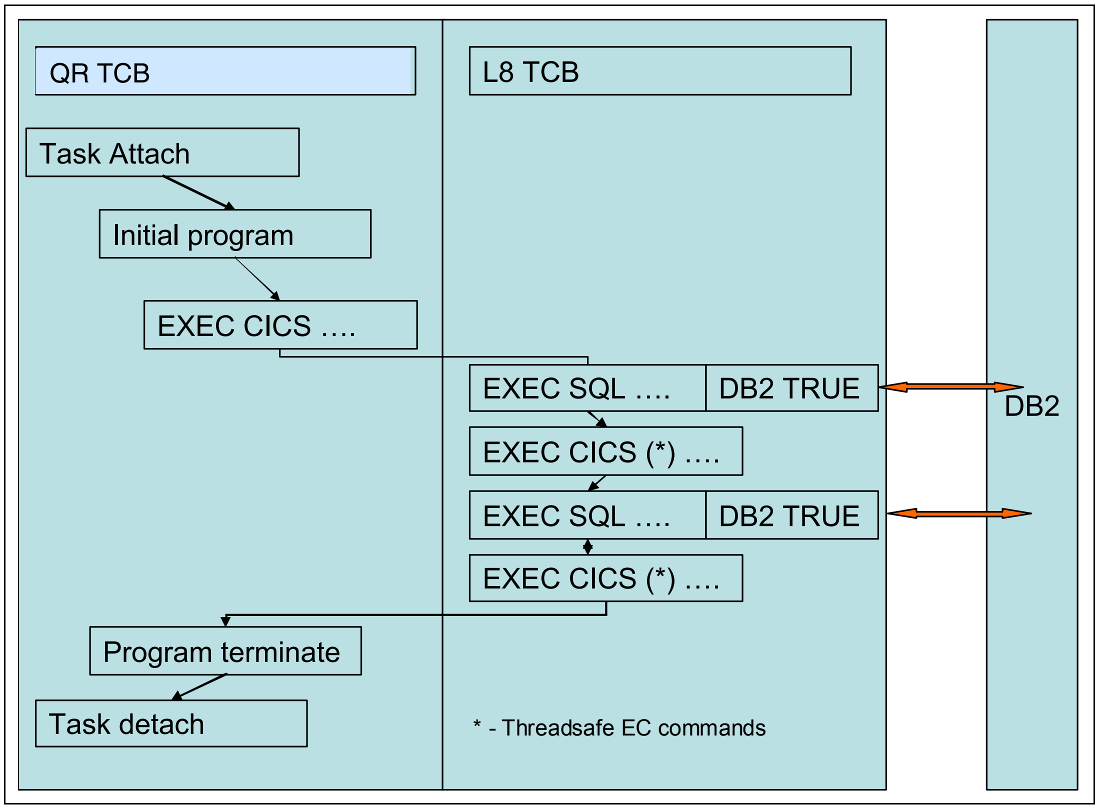


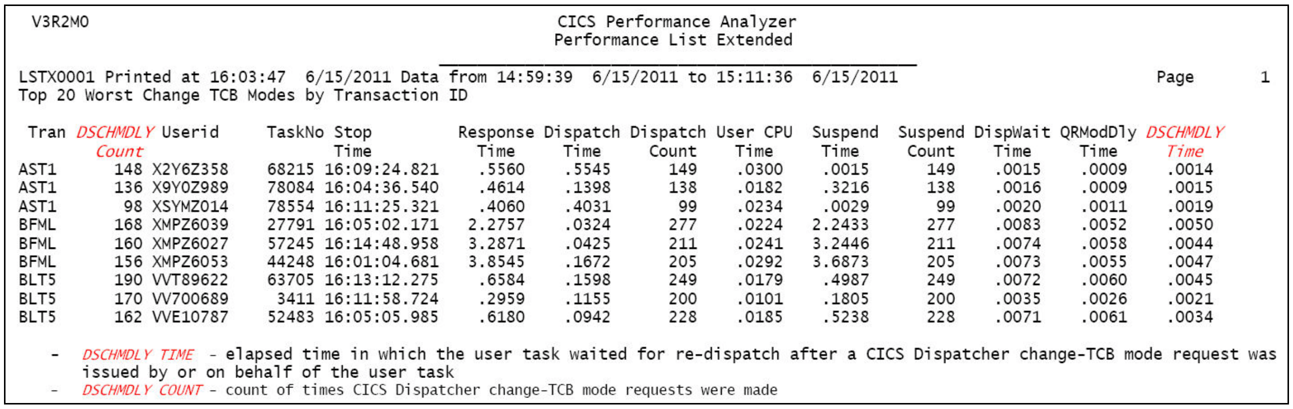
Figure 7 Typical TCB use for threadsafe CICS DB2 transaction

End-of-task sync point processing: In CICS Transaction Server V4.2, the end-of-task sync point processing occurs, if possible, on the L8 TCB, although in same cases processing can be forced onto the QR TCB. However, in all cases, end-of-task processing occurs on the QR TCB.

In this example, two task switches are saved. In a typical customer application, more SQL activity occurs, which can result in a CPU savings if programs are coded and defined as threadsafe transactions.

Parameter note: In CICS Transaction Server V4.2, the combination of the **`CONCURRENCY(REQUIRED)`** and **`API(CICSAPI)`** settings results in starting the initial application program on an L8 TCB. Whether you use these parameters or the **`CONCURRENCY(THREADSAFE) API(CICSAPI)`** parameter depends on how many nonthreadsafe commands the program issues.

CICS Performance Analyzer provides a sample report, BADCHMDS, of the top 20 most frequent task switches by transaction. You can view this report, and other reports, in the sample report forms option of the CICS Performance Analyzer ISPF interface. Figure 8 shows a sample of the output of this report.



```
 V3R2M0                                                     CICS Performance Analyzer
                                                            Performance List Extended
                                                   ____________________________________________
LSTX0001 Printed at 16:03:47  6/15/2011 Data from 14:59:39  6/15/2011 to 15:11:36  6/15/2011                     Page       1
Top 20 Worst Change TCB Modes by Transaction ID

 Tran DSCHMDLY Userid      TaskNo Stop          Response Dispatch Dispatch User CPU  Suspend  Suspend DispWait QRModDly DSCHMDLY
        Count                     Time            Time     Time     Count    Time     Time     Count    Time     Time     Time
AST1       148 X2Y6Z358    68215 16:09:24.821   .5560    .5545      149    .0300    .0015      149    .0015    .0009    .0014
AST1       136 X9Y0Z989    78084 16:04:36.540   .4614    .1398      138    .0182    .3216      138    .0016    .0009    .0015
AST1        98 XSYMZ014    78554 16:11:25.321   .4060    .4031       99    .0234    .0029       99    .0020    .0011    .0019
BFML       168 XMPZ6039    27791 16:05:02.171  2.2757    .0324      277    .0224   2.2433      277    .0083    .0052    .0050
BFML       160 XMPZ6027    57245 16:14:48.958  3.2871    .0425      211    .0241   3.2446      211    .0074    .0058    .0044
BFML       156 XMPZ6053    44248 16:01:04.681  3.8545    .1672      205    .0292   3.6873      205    .0073    .0055    .0047
BLT5       190 VVT89622    63705 16:13:12.275   .6584    .1598      249    .0179    .4987      249    .0072    .0060    .0045
BLT5       170 VV700689     3411 16:11:58.724   .2959    .1155      200    .0101    .1805      200    .0035    .0026    .0021
BLT5       162 VVE10787    52483 16:05:05.985   .6180    .0942      228    .0185    .5238      228    .0071    .0061    .0034

    -   DSCHMDLY TIME  - elapsed time in which the user task waited for re-dispatch after a CICS Dispatcher change-TCB mode request was
        issued by or on behalf of the user task
    -   DSCHMDLY COUNT - count of times CICS Dispatcher change-TCB mode requests were made
```

Figure 8   Sample output from CICS Performance Analyzer report on the top 20 worst change TCB modes

For more information about the use of OTE for CICS-DB2 applications, see "CICS threadsafe considerations" on page 17.

Thread allocation and thread protection

A cost is associated with the allocation and deallocation of a thread to DB2. Thus, it is beneficial to optimize the CICS DB2 interface so that thread reuse occurs. All pool and entry threads are eligible for reuse if the PLAN that is used remains unchanged. For the pool and unprotected entry threads, this reuse occurs only if a transaction is waiting for a thread at the time the thread might normally be terminated.

For this reason, protected threads were introduced. The PROTECTNUM attribute of the DB2ENTRY definition specifies the maximum number of threads that are placed in a wait state, rather than deallocated, if no transaction is immediately available to reuse it. The protected thread is kept in an idle state for an average of two purge cycles. The duration is specified by the PURGECYCLE attribute on the DB2CONN definition, at which time it is terminated.

A performance benchmark was undertaken to show the CPU cost comparison of `ALLOCATE` and `DEALLOCATE` versus thread reuse.

The RTW workload was run at different rates with the **`PROTECTNUM=0`** setting specified on the DB2ENTRY definition. Observations showed that no thread reuse was occurring. The workloads were then repeated with the **`PROTECTNUM=5`** setting specified. In this example, statistics reports indicated that thread reuse was occurring.

It was observed that thread reuse (through the specification of protected threads) resulted in lower CPU usage and reduced response times. The average CPU time per transaction with the **`PROTECTNUM=0`** setting was 3.5 milliseconds, and the average CPU time per transaction with the **`PROTECTNUM=5`** setting was 3.2 milliseconds.

CICS Transaction Server V4.2 includes the following changes:

- The minimum allowable value of the PURGECYCLE attribute is reduced to 5 seconds.
- A REUSELIMIT attribute on the DB2CONN definition controls the maximum number of times a thread can be reused before it is forced to terminate. The range 0 - 10,000 is allowed with a default of 1,000. This attribute was introduced to reduce the over allocation of thread and environmental descriptor manager (EDM) pool storage for packages and plans that specify the `RELEASE(DEALLOCATE)` BIND option.

When using thread allocation and thread protection, keep in mind the following notes:

- The REUSELIMIT default value can solicit a change in existing CICS behavior.
- CPU costs listed are processor-dependent. Use the LSPR metrics to estimate the cost that is incurred on different processor models and operating system levels.
- A general rule of thumb is that savings incurred with thread reuse are 0% - 10% of CPU cost per transaction. The cost of thread creation or termination is not reflected in the DB2 accounting records (SMF type 101). Analyze the CICS monitoring records (SMF 110) to determine the cost savings that are achieved.
- Savings between the `RELEASE(DEALLOCATE)` and `RELEASE(COMMIT)` settings are achieved only if thread reuse is occurring.

Thread priority

A benchmark was performed to measure the difference in CPU time and transaction throughput with differing values specified for the `PRIORITY` attribute.

The RTW workload was set to run with the `CONCURRENCY(THREADSAFE)`, `API(OPENAPI)`, and `STGPROTECT=NO` settings. These settings ensure that the transaction is running on an L8 TCB before the first SQL call being issued. No significant difference in CPU time or transaction throughput was observed running with the `PRIORITY(HIGH)`, `PRIORITY(EQUAL)`, or `PRIORITY(LOW)` settings.

The RTW workload was altered to run with the `CONCURRENCY(QUASIRENT)`, `API(CICSAPI)`, and `STGPROTECT=NO` settings. Greater throughput was observed with the `PRIORITY(HIGH)` setting, with no noticeable difference in CPU usage with the `PRIORITY(HIGH)`, `PRIORITY(EQUAL)`, or `PRIORITY(LOW)` settings.

CICS threadsafe applications that issue DB2 calls by using `PRIORITY(HIGH)` threads will have L8 TCBs that remain at the higher priority from the first SQL call until the transaction terminates. This higher priority can lead the QR TCB to starve from the necessary CPU resource because the L8 TCBs dominate CPU usage. Review the CICS Statistics report for each CICS region regularly to ensure that the dispatch to CPU ratio for the QR TCB does not fall below 80%. Although the default setting is `PRIORITY(HIGH)`, tune the value for your specific environment.

DB2 BIND options

Figure 1 on page 2 shows the suggested thread types and BIND options to use for differing transaction types. This information provides a good starting point from which thorough measurement and analysis can be used to tune your environment.

The general rule is that for high frequency transactions, where thread reuse is likely to occur, use the `ACQUIRE(ALLOCATE)` and `RELEASE(DEALLOCATE)` BIND options. For transactions where thread reuse is less likely or where transaction frequency is low or sporadic, use the **ACQUIRE(USE)** and **RELEASE(COMMIT)** BIND options.

The two main factors that contribute to this strategy are thread allocation costs and DB2 EDM pool resource usage.

For the **ACQUIRE(ALLOCATE)** and **RELEASE(DEALLOCATE)** options, 31-bit EDM virtual storage usage can become high, especially with plans where package content is high and thread reuse by differing transactions is common. To alleviate this high virtual storage use, CICS Transaction Server V4.2 includes the **REUSELIMIT** parameter. In addition, in DB2 V10 the storage for the EDM pool is moved into 64-bit storage.

DB2 V10 and the ACQUIRE(ALLOCATE) parameter: The **ACQUIRE(ALLOCATE)** parameter is not supported in DB2 V10 and later.

To use the EDM pool storage above the bar:

1. Rebind packages and plans by using the V10 level code.
2. Define DB2 buffer pools with the **PGFIX=YES** option (only in production systems where sufficient real storage is available to avoid paging to auxiliary storage).
3. Stop and start the DB2 subsystems.
4. Define the z/OS **LFAREA** parameter to accommodate the real storage requirements. For DB2, this definition involves calculating the sum of the DB2 buffer pool sizes that are defined with the **PGFIX(YES)** option.

For more information, see *DB2 10 for z/OS Managing Performance*, SC19-2978.

Storage above the 2 GB bar: Following these steps in V10 can help you to achieve in excess of 95% of thread-related storage that is above the 2 GB bar.

Additionally, for performance purposes, include the collections for the most frequently used programs first in the package list when binding DB2 plans for CICS DB2 applications. The reason is because the collections are searched in the order specified to locate the DBRM. DB2 statistics show the number of searches where DBRMS were not located.

A performance benchmark was undertaken to show the differences in CPU cost incurred by transactions with differing options and thread reuse.

The RTW workload was run in a steady state against threads that were defined to a DB2ENTRY, which was altered to force the following characteristics. The list numbers correspond to the explanation that follows this list.

1. For 0 protected threads, **BIND RELEASE(COMMIT)** recorded CPU time per transaction of 3.5 milliseconds with 0% thread reuse.
2. For 5 protected threads, **BIND RELEASE(COMMIT)** recorded CPU time per transaction of 3.2 milliseconds with 100% thread reuse.
3. For 0 protected threads, **BIND RELEASE(DEALLOCATE)** recorded CPU time per transaction of 3.5 milliseconds with 0% reuse.
4. For 5 protected threads, **BIND RELEASE(DEALLOCATE)** recorded CPU time per transaction of 2.6 milliseconds with 100% thread reuse.

Keep in mind the following notes:

- Results 1 and 3 force a similar behavior because no thread reuse results in plan deallocation at transaction termination (with implied commit). Thus, **`RELEASE(COMMIT)`** and **`RELEASE(DEALLOCATE)`** are the same.
- The difference between results 1 and 2 is the difference in cost between thread reuse and thread allocate or deallocate.
- The difference between result 2 and 4 is the difference between DB2 package and plan reuse over allocation and deallocation. Result 4 will not result in releasing EDM pool content at transaction termination (implied commit) due to 100% thread reuse.

An additional factor that can have a considerable affect on performance is the number of packages that are defined to the plan. Many IBM customers have adopted a strategy of running single DB2 SQL statements per program, which leads to plans with high numbers of packages.

A benchmark was taken to show the difference in CPU usage between the following programs:

- Program A was defined to access 300 DB2 tables that use cursors (OPEN, FETCH, and CLOSE).
- Program B was defined to statically call 300 individual subprograms, where each subprogram is defined to access a single DB2 table that uses a cursor (OPEN, FETCH, and CLOSE).

The result was the following plans:

DBL1	A plan that contains a single large package for all 300 table accesses.
DBM1	A plan that contains 300 small packages, one for each table that is accessed.

Each program was run in a steady state workload with the CPU cost per transaction measured as follows:

- DBL1 used 54.2 milliseconds of CPU per transaction.
- DBM1 used 72.3 milliseconds of CPU per transactions (representing a 33% increase in CPU).

An additional factor that affects performance is PKLIST. Keep the list of collection as short as possible, with the order of collections strictly prioritized based on frequency of package access. Also, monitor the ratio of package allocate failures to package allocation successes.

DB2 BIND ISOLATION option

The following **`ISOLATE`** options are allowable on the DB2 **`BIND`** command:

UR	Uncommitted Read
CS	Cursor Stability
RS	Read Stability
RR	Repeatable Read

The CS option allows maximum concurrency with data integrity.

For **`ISOLATION(CS)`**, use the **`CURRENTDATA(NO)`** option as a design default to allow maximum concurrency and maximum lock avoidance. However, to fully understand the implications of this option, see *DB2 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

CICS threadsafe considerations

Write applications that are developed to run in the CICS environment to threadsafe standards to allow the application to use the OTE and benefit from the parallelism that an OTE provides. In addition to this benefit, CPU savings can be achieved by certain applications, particularly CICS DB2 applications where the interaction with DB2 always occurs on an OTE L8 TCB.

In addition to writing applications to threadsafe standards, several environmental factors that are controlled by key CICS definition settings control when and under which TCB programs will run.

Table 2 summarizes the TCB under which an application runs, depending on the system and program settings that are listed in Table 1.

Table 1 System and program settings

Option	Meaning
STGPROT	Storage protection active within a CICS region
EXECKEY	Execution key of the program: CICSKEY or USERKEY
CONCURRENCY	THREADSAFE, QUASIRENT, or REQUIRED[a]
API	CICS or OPEN

a. REQUIRED is new setting in CICS Transaction Server V4.2

Table 2 CICS Transaction Server V4.2 TCB usage based on application resource usage

STGPROT	Exec key	Concurrency	API	Initial TCB	DB2 or MQ command	Threadsafe command	Nonthreadsafe command
Yes/No	Any	Quasi	CICS	QR	QR/L8/QR	No change	No change
Yes/No	Any	Threadsafe	CICS	QR	L8	No change	QR
Yes/No	Any	Required	CICS	L8	No change	No change	L8/QR/L8
No	Any	Threadsafe	Open	L8	No change	No change	L8/QR/L8
No	Any	Required	Open	L8	No change	No change	L8/QR/L8
Yes	CICS	Threadsafe	Open	L8	No change	No change	L8/QR/L8
Yes	CICS	Required	Open	L8	No change	No change	L8/QR/L8
Yes	User	Threadsafe	Open	L9	L9/L8/L9	No change	L9/QR/L9
Yes	User	Required	Open	L9	L9/L8/L9	No change	L9/QR/L9

In all cases, the task attach and detach processing is performed on the QR TCB. The INITIAL TCB column refers to the TCB under which the first or initial application program (as defined in the transaction definition) runs when starting the RTW workload with different EXECKEY, CONCURRENCY, and API settings. Table 3 lists the results.

Table 3 Results

Concurrency	API	STGPROT	EXECKEY	Average CPU per transaction
QUASIRENT	CICS	N/A	USER	5.7 milliseconds
THREADSAFE	CICS	N/A	USER	3.5 milliseconds
THREADSAFE	OPEN	NO	USER	3.5 milliseconds
THREADSAFE	OPEN	YES	USER	6.1 milliseconds

Higher TCB switching: In CICS Transaction Server V4.2, using the `API(CICSAPI)` and `CONCURRENCY(REQUIRED)` parameters with programs that issue nonthreadsafe EXEC CICS commands can result in a higher amount of TCB switching than applications that specify the `API(CICSAPI)` and `CONCURRENCY(THREADSAFE)` parameters. The reason is that CICS for `CONCURRENCY(THREADSAFE)` keeps the program running on the QR TCB between two consecutive nonthreadsafe EXEC CICS commands. With the `CONCURRENCY(REQUIRED)` parameter, CICS switches back to the L8 TCB between the commands.

In Table 3, the considerable increase in CPU for the `API(OPEN) EXECKEY(USER)` transactions is a result of the transaction that is running on an L9 TCB for all processing other than the processing performed by DB2.

For most CICS-DB2 transactions, define the programs as `API(CICS)` and `EXECKEY(USER)`.

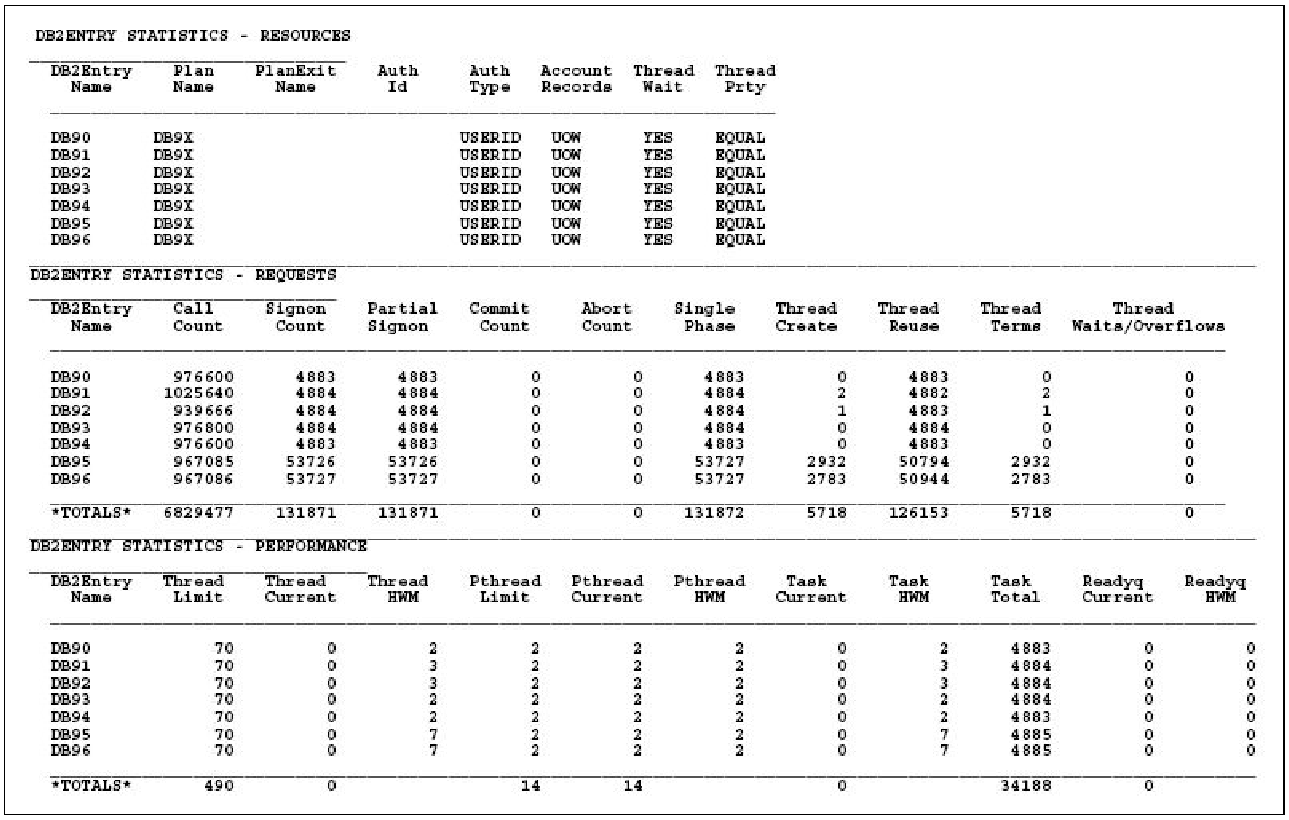
For applications that contain a low number of nonthreadsafe commands, using the `CONCURRENCY(REQUIRED)` parameter can provide benefits in terms of throughput because the application is moved to the open TCB earlier in its processing.

CICS statistics and monitoring

The CICS DFHSTUP statistics program formats CICS SMF 110 subtype 2 records to produce a report that shows activity for various components of a CICS region. The `SELECT TYPE=(DB2)` statement generates a report that shows the DB2 CONNECTION information and activity with DB2ENTRY resource, requests, and performance information. This report, with an appropriate statistics collection interval, is a useful tool in the analysis of CICS DB2 activity.

Parameters: CICS statistics collection is controlled by the `STATRCD=(YES|NO)`, `STATEOD=(0,HHMMSS)` and `STATINT=HHMMSS DFHSIT` parameters.

The CICS statistics reports shown in Figure 9 highlight the effect of several of the attributes mentioned in this paper.



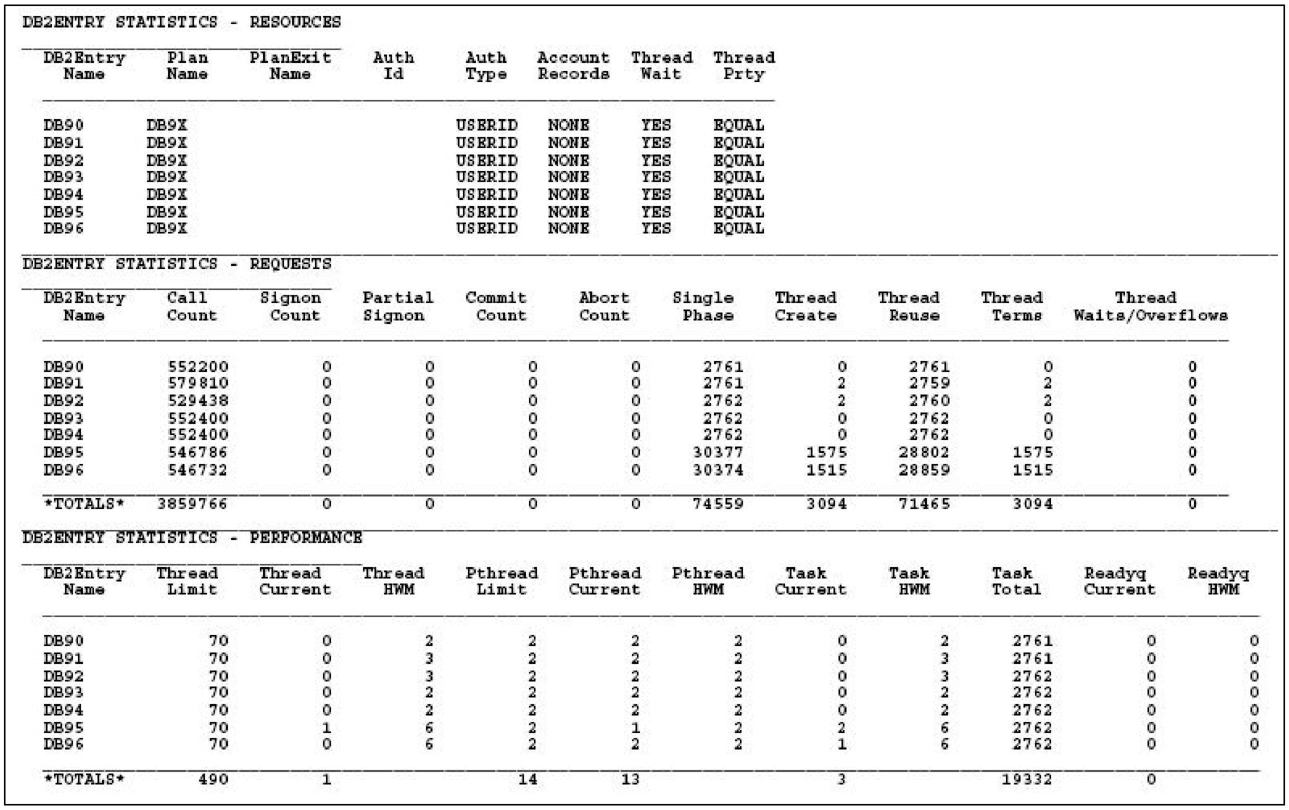
```
DB2ENTRY STATISTICS - RESOURCES
________________________________________________________________________________
 DB2Entry    Plan    PlanExit    Auth      Auth    Account   Thread   Thread
   Name      Name      Name       Id       Type    Records    Wait     Prty
 ______________________________________________________________________________

 DB90      DB9X                            USERID   UOW        YES      EQUAL
 DB91      DB9X                            USERID   UOW        YES      EQUAL
 DB92      DB9X                            USERID   UOW        YES      EQUAL
 DB93      DB9X                            USERID   UOW        YES      EQUAL
 DB94      DB9X                            USERID   UOW        YES      EQUAL
 DB95      DB9X                            USERID   UOW        YES      EQUAL
 DB96      DB9X                            USERID   UOW        YES      EQUAL
DB2ENTRY STATISTICS - REQUESTS
________________________________
 DB2Entry     Call    Signon   Partial   Commit    Abort   Single   Thread   Thread   Thread      Thread
   Name      Count     Count    Signon    Count    Count    Phase   Create    Reuse    Terms  Waits/Overflows
 _____________________________________________________________________________________________________________

 DB90       976600      4883      4883        0        0     4883        0     4883        0              0
 DB91      1025640      4884      4884        0        0     4884        2     4882        2              0
 DB92       939666      4884      4884        0        0     4884        1     4883        1              0
 DB93       976800      4884      4884        0        0     4884        0     4884        0              0
 DB94       976600      4883      4883        0        0     4883        0     4883        0              0
 DB95       967085     53726     53726        0        0    53727     2932    50794     2932              0
 DB96       967086     53727     53727        0        0    53727     2783    50944     2783              0
 _____________________________________________________________________________________________________________
 *TOTALS*  6829477    131871    131871        0        0   131872     5718   126153     5718              0
DB2ENTRY STATISTICS - PERFORMANCE
__________________________________
 DB2Entry   Thread    Thread   Thread   Pthread   Pthread   Pthread    Task     Task     Task    Readyq    Readyq
   Name      Limit   Current     HWM     Limit    Current      HWM   Current     HWM    Total   Current      HWM
 _______________________________________________________________________________________________________________

 DB90           70         0        2        2         2         2        0        2     4883         0         0
 DB91           70         0        3        2         2         2        0        3     4884         0         0
 DB92           70         0        3        2         2         2        0        3     4884         0         0
 DB93           70         0        2        2         2         2        0        2     4884         0         0
 DB94           70         0        2        2         2         2        0        2     4883         0         0
 DB95           70         0        7        2         2         2        0        7     4885         0         0
 DB96           70         0        7        2         2         2        0        7     4885         0         0
 _______________________________________________________________________________________________________________
 *TOTALS*      490         0                14        14                  0              34188         0
```

Figure 9 Sample CICS statistics report 1

Figure 9 shows the following information:

- ACCOUNTREC(UOW) sign-on and partial sign-on counts are the same, indicating that CICS issued only partial sign-ons. DB2 accounting records are created each time a UOW is ended.
- Thread high water mark (HWM) for DB90 through DB94 shows high allocation and reuse of protected threads. This result is also shown in the Thread HWM and pthread HWM of the DB2ENTRY STATISTICS - PERFORMANCE report with reuse shown in the Thread Reuse column of the DB2ENTRY - REQUESTS report.
- Compare this information with transaction DB95 and DB96, where insufficient protected threads are assigned, as shown in HWM Settings, resulting in an increase in thread creates and terminates.

Figure 10 is a statistics report from a similar workload to the workload that generated report 1 in Figure 9 on page 19.



```
DB2ENTRY STATISTICS - RESOURCES
________________________________________________________________________________
 DB2Entry    Plan     PlanExit    Auth      Auth   Account  Thread  Thread
   Name      Name       Name       Id       Type   Records   Wait    Prty
 ______________________________________________________________________________

 DB90       DB9X                           USERID   NONE     YES    EQUAL
 DB91       DB9X                           USERID   NONE     YES    EQUAL
 DB92       DB9X                           USERID   NONE     YES    EQUAL
 DB93       DB9X                           USERID   NONE     YES    EQUAL
 DB94       DB9X                           USERID   NONE     YES    EQUAL
 DB95       DB9X                           USERID   NONE     YES    EQUAL
 DB96       DB9X                           USERID   NONE     YES    EQUAL
_________________________________________________________________________________________________________________________________
DB2ENTRY STATISTICS - REQUESTS
_______________________________
 DB2Entry    Call     Signon    Partial   Commit    Abort    Single   Thread   Thread   Thread       Thread
   Name      Count     Count    Signon     Count    Count     Phase   Create    Reuse    Terms  Waits/Overflows
 _____________________________________________________________________________________________________________

 DB90       552200         0         0         0        0      2761        0     2761        0                0
 DB91       579810         0         0         0        0      2761        2     2759        2                0
 DB92       529438         0         0         0        0      2762        2     2760        2                0
 DB93       552400         0         0         0        0      2762        0     2762        0                0
 DB94       552400         0         0         0        0      2762        0     2762        0                0
 DB95       546786         0         0         0        0     30377     1575    28802     1575                0
 DB96       546732         0         0         0        0     30374     1515    28859     1515                0
 _____________________________________________________________________________________________________________
 *TOTALS*  3859766         0         0         0        0     74559     3094    71465     3094                0
_________________________________________________________________________________________________________________________________
DB2ENTRY STATISTICS - PERFORMANCE
_________________________________
 DB2Entry  Thread    Thread    Thread   Pthread  Pthread  Pthread    Task     Task     Task   Readyq   Readyq
   Name     Limit   Current     HWM      Limit   Current    HWM    Current    HWM     Total  Current    HWM
 _______________________________________________________________________________________________________________

 DB90          70         0         2        2        2        2        0        2     2761        0        0
 DB91          70         0         3        2        2        2        0        3     2761        0        0
 DB92          70         0         3        2        2        2        0        3     2762        0        0
 DB93          70         0         2        2        2        2        0        2     2762        0        0
 DB94          70         0         2        2        2        2        0        2     2762        0        0
 DB95          70         1         6        2        1        2        2        6     2762        0        0
 DB96          70         0         6        2        2        2        1        6     2762        0        0
 _______________________________________________________________________________________________________________
 *TOTALS*     490         1                 14       13                 3              19332        0
```

Figure 10 CICS Sample Statistics report 2

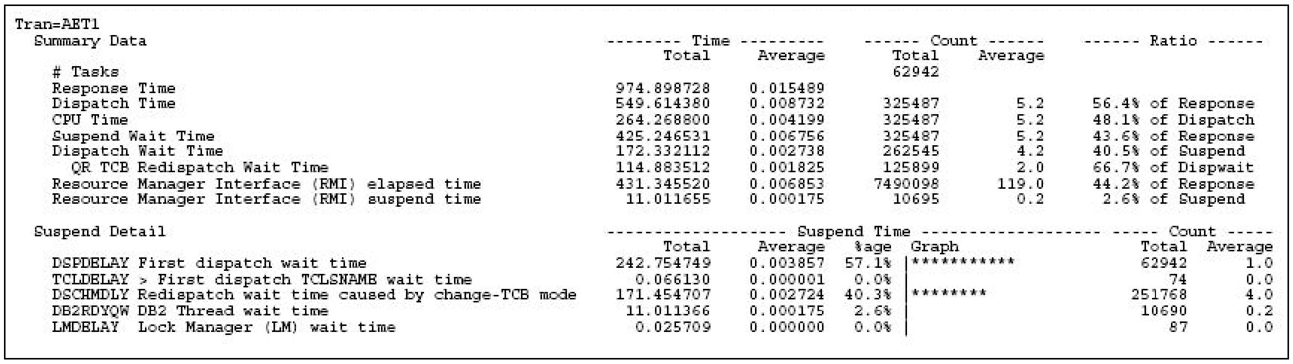
This example includes the following information:

- The DB2ENTRY definition is defined with **ACCOUNTREC(NONE)**. The sign-on and partial sign-on counts are zero because no DB2 accounting records are being forced by CICS.
- Thread allocation and reuse are similar to thread allocation and reuse in report 1.

CICS Performance Analyzer is frequently used instead of the CICS supplied program **DFH$MOLS** to format and print the contents of the CICS SMF 110 subtype 1 monitoring records.

CICS Performance Analyzer provides several reports that are useful in understanding the CICS DB2 activity that occurs within a CICS region or individual transaction.

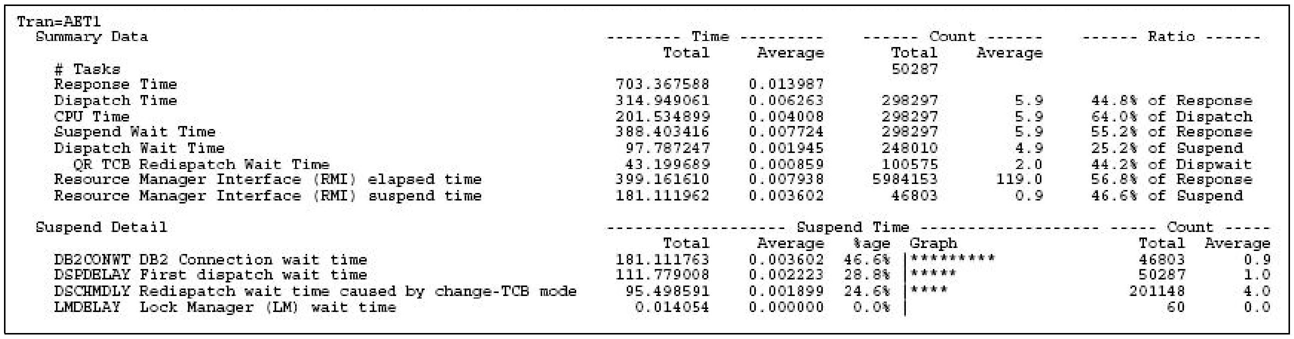
The reports in Figure 11 and Figure 12 show the effect of alterations to key attributes and some of the unexpected side affects of the changes. Figure 11 shows the wait analysis report for a transaction that opens a DB2 cursor, retrieves 100 rows, and closes the cursor. The DB2 **CTHREAD** parameter is set to 20, the CICS **DB2CONN TCBLIMIT** is set to 40, and the **DB2ENTRY** definition specifies a **THREADLIMIT** of 10 with no protected threads. A workload with 10 clients was run.



```
Tran=AET1
  Summary Data                                                -------- Time ---------    ------ Count ------    ------ Ratio ------
                                                                   Total     Average       Total   Average
    # Tasks                                                                                62942
    Response Time                                             974.898728    0.015489
    Dispatch Time                                             549.614380    0.008732      325487       5.2     56.4% of Response
    CPU Time                                                  264.268800    0.004199      325487       5.2     48.1% of Dispatch
    Suspend Wait Time                                         425.246531    0.006756      325487       5.2     43.6% of Response
    Dispatch Wait Time                                        172.332112    0.002738      262545       4.2     40.5% of Suspend
      QR TCB Redispatch Wait Time                             114.883512    0.001825      125899       2.0     66.7% of Dispwait
    Resource Manager Interface (RMI) elapsed time             431.345520    0.006853     7490098     119.0     44.2% of Response
    Resource Manager Interface (RMI) suspend time              11.011655    0.000175       10695       0.2      2.6% of Suspend

  Suspend Detail                                              ------------------ Suspend Time ------------------ ----- Count -----
                                                                   Total     Average    %age  Graph                  Total  Average
    DSPDELAY First dispatch wait time                         242.754749    0.003857   57.1% |***********            62942      1.0
    TCLDELAY > First dispatch TCLSNAME wait time                0.066130    0.000001    0.0% |                          74      0.0
    DSCHMDLY Redispatch wait time caused by change-TCB mode   171.454707    0.002724   40.3% |********              251768      4.0
    DB2RDYQW DB2 Thread wait time                              11.011366    0.000175    2.6% |                       10690      0.2
    LMDELAY  Lock Manager (LM) wait time                        0.025709    0.000000    0.0% |                          87      0.0
```

Figure 11 DB2 wait analysis showing no forced waits and no protected threads

Figure 12 shows the effect of reducing the **TCBLIMIT** on the **DB2CONN** definition. The DB2 **CTHREAD** setting remains at 20, but the CICS **TCBLIMIT** on the **DB2CONN** definition is lowered to 4. The DB2ENTRY settings remain the same. The report shows a wait of type DB2CONWT occurring as the throughput of transactions is throttled by the **TCBLIMIT** change.



```
Tran=AET1
  Summary Data                                                -------- Time ---------    ------ Count ------    ------ Ratio ------
                                                                   Total     Average       Total   Average
    # Tasks                                                                                50287
    Response Time                                             703.367588    0.013987
    Dispatch Time                                             314.949061    0.006263      298297       5.9     44.8% of Response
    CPU Time                                                  201.534899    0.004008      298297       5.9     64.0% of Dispatch
    Suspend Wait Time                                         388.403416    0.007724      298297       5.9     55.2% of Response
    Dispatch Wait Time                                         97.787247    0.001945      248010       4.9     25.2% of Suspend
      QR TCB Redispatch Wait Time                              43.199689    0.000859      100575       2.0     44.2% of Dispwait
    Resource Manager Interface (RMI) elapsed time             399.161610    0.007938     5984153     119.0     56.8% of Response
    Resource Manager Interface (RMI) suspend time             181.111962    0.003602       46803       0.9     46.6% of Suspend

  Suspend Detail                                              ------------------ Suspend Time ------------------ ----- Count -----
                                                                   Total     Average    %age  Graph                  Total  Average
    DB2CONWT DB2 Connection wait time                         181.111763    0.003602   46.6% |*********              46803      0.9
    DSPDELAY First dispatch wait time                         111.779008    0.002223   28.8% |*****                  50287      1.0
    DSCHMDLY Redispatch wait time caused by change-TCB mode    95.498591    0.001899   24.6% |****                  201148      4.0
    LMDELAY  Lock Manager (LM) wait time                        0.014054    0.000000    0.0% |                          60      0.0
```

Figure 12 DB2 wait analysis with TCBLIMIT lowered to force a DB2CONWT delay

Similarly, if the THREADLIMIT is set such that the throttling occurs on the thread allocation, a wait of DB2RDYQW occurs.

DB2 accounting, statistics, and commands

This section provides a brief overview of the accounting and statistics data that is generated by DB2. It explains how CICS can control the amount of data generated, by using the ACCOUNTREC attribute of the CICS DB2 **DB2CONN** and **DB2ENTRY** definitions. It also includes sample job control language (JCL) to format the data along with extracts from the reports that are created. Finally, an overview of the DB2 **DISPLAY THREAD** command shows how CICS-DB2 connection information can be displayed.

DB2 accounting

Although CICS monitoring and statistics give a CICS perspective of the DB2 activity that is occurring, the most detailed information is gathered with DB2 itself. The DB2 accounting trace is split into several accounting classes, which must be active for the entirety of the transaction for a complete picture to be available. For more information, see the *DB2 for z/OS Performance Monitoring and Tuning Guide*, SC18-9851.

Figure 13 illustrates the breakdown of response time from a DB2 perspective.

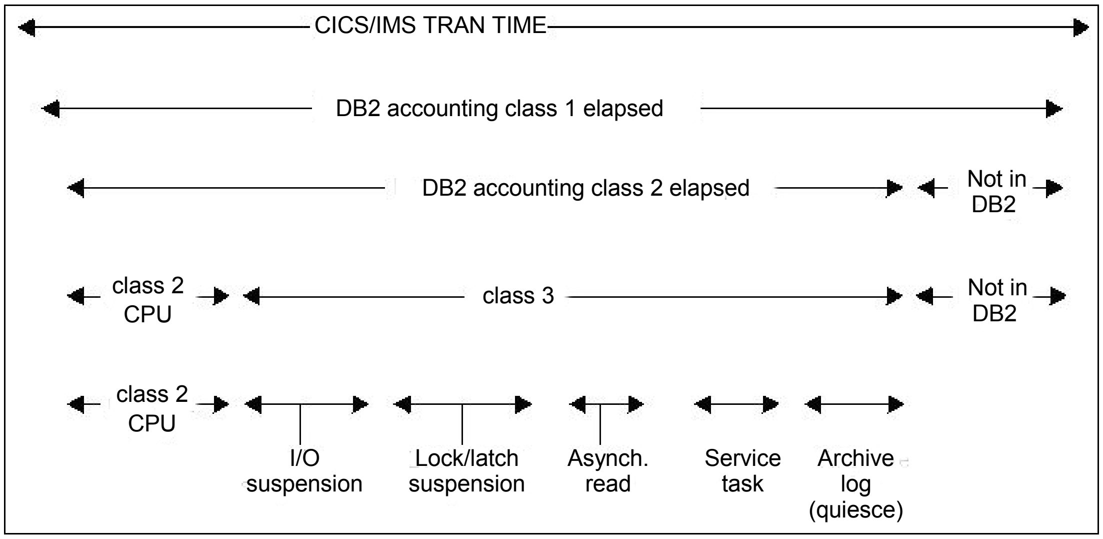


Figure 13 DB2 response time breakdown

The response time from a DB2 perspective has the following breakdown:

- DB2 accounting elapse times

  These times are collected in the accounting records. They are taken over the accounting interval between the point where DB2 starts to run the first SQL statement, and the point that precedes thread termination or reuse by a different user (*Signon*). The interval excludes the time spent in creating and terminating a thread.

  DB2 records the following elapse times:

  – *Class 1 elapsed time* is always presented in the accounting records and shows the duration of the accounting interval. It includes the time spent in DB2 and the time spent in the front end. With CICS protected threads where reuse is occurring, the Class 1 elapsed time is the time period between successive sign-ons. It includes the time when the thread is idle and waiting for a new CICS transaction to arrive and reuse the thread.

  – *Class 2 elapsed time* is produced only if the accounting class 2 trace is active and counts only the time spent in the DB2 address space during the accounting interval. It represents the sum of the times from any entry into DB2 until the corresponding exit from DB2. It is also referred to as the time spent in DB2. If class 2 is not active for the duration of the thread, the class 2 elapse time might not reflect the entire DB2 time for the thread, but only the time when the class 2 trace was active.

  – *Class 3 elapsed time* is produced only if the accounting class 3 trace is active and is the elapsed wait time in DB2. Possible causes of the wait time include synchronous I/O suspension time, lock/latch suspension time, asynchronous read suspensions, asynchronous write suspensions, service task suspensions, and archive log mode time.

- DB2 total transit time

  For an SQL transaction or query, the total transit time is the elapsed time from the beginning of create thread, or the sign-on of another authorization ID when reusing the thread, until the end of thread termination, or the sign-on of another authorization ID.

  When a thread is reused, the transit time can include idle time. Unlike the accounting class 1 and 2 times, it includes the time spent in the create and terminate thread.

For CICS, the frequency of accounting record generation depends on several factors. Under all circumstances, assuming DB2 accounting trace is active, DB2 generates accounting records when the authorization ID of the thread changes or the thread is terminated. In addition to these records, through the **DB2CONN** and **DB2ENTRY ACCOUNTREC** attribute settings, CICS can control the conditions under which DB2 generates additional records.

You might choose a particular setting for many reasons. Performance might not be the overriding consideration in the choice made. However, from a performance perspective, the RTW workload was run with differing **ACCOUNTREC** settings. Approximately 16,500 transactions were started per run with the results shown in Table 4.

Table 4 Results of the RTW workload run with differing ACCOUNTREC settings

ACCOUNTREC setting	**DB2 accounting records generated**	**Average CPU cost per transaction**
TASK	150,000	3.4 milliseconds
UOW	330,000	3.5 milliseconds
NONE	12,000	3.29 milliseconds

For Table 4, the following DB2 accounting records are generated:

- ACCOUNTREC(NONE)

  Twelve thousand records were created because some thread reuse occurred. The number of thread terminations for the 16,500 transactions was approximately 12,000.

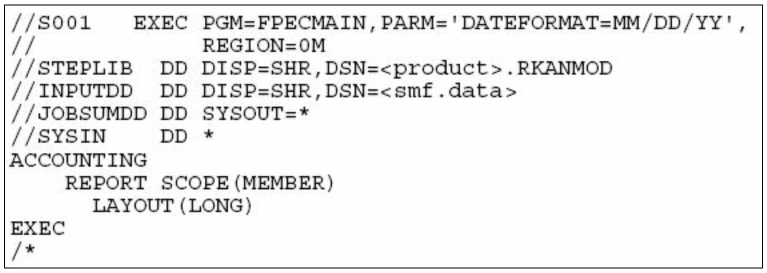
- ACCOUNTREC(TASK)

  One hundred and fifty thousand records were created because a subset of the transactions in the workload contains multiple UOWs. After the synchronization point, if a new thread was used to continue the transaction activity, an accounting record is generated. Therefore, you can expect to see one record per task termination, plus one record for each new thread that is used by the transaction.

- ACCOUNTREC(UOW)

  Three hundred and thirty thousand records were created because a subset of the transactions in the workload perform multiple sync points. For each sync point and at task termination, a record is generated.

To format the DB2 accounting records, you can use IBM OMEGAMON XE DB2 Performance Expert for z/OS. Figure 14 shows the sample JCL used to format the DB2 accounting records.



```
//S001   EXEC PGM=FPECMAIN,PARM='DATEFORMAT=MM/DD/YY',
//            REGION=0M
//STEPLIB  DD DISP=SHR,DSN=<product>.RKANMOD
//INPUTDD  DD DISP=SHR,DSN=<smf.data>
//JOBSUMDD DD SYSOUT=*
//SYSIN    DD *
ACCOUNTING
    REPORT SCOPE(MEMBER)
      LAYOUT(LONG)
EXEC
/*
```

Figure 14   DB2 Performance Expert sample JCL for accounting report

Important: Use DB2 accounting trace classes 1, 2, 3, 7, and 8 as the design default. Classes 7 and 8 provide package-level accounting.

DB2 statistics

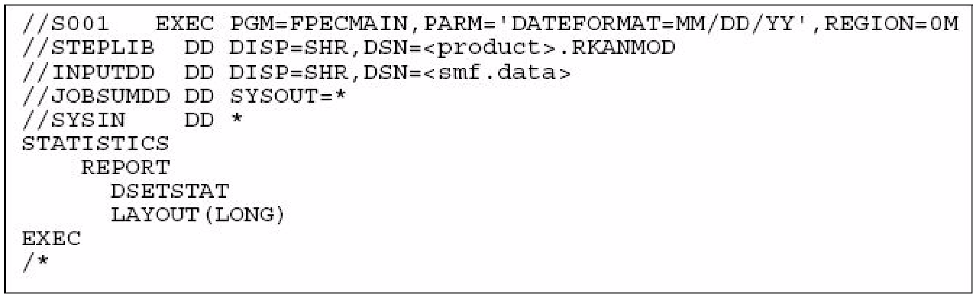
Similar to CICS, DB2 collects statistics data at defined intervals. Similar to DB2 accounting records, the trace classes that are activated determine the data that is collected.

The following trace classes are available:

Class 1	Provides information about system services, database statistics, and statistics for the DBM1 address space. It also includes the system parameters that were in effect when the trace was started.
Class 2	Provides installation-defined statistics records.
Class 3	Provides information about deadlocks and timeouts.
Class 4	Provides information about exceptional conditions.
Class 5	Provides information about data sharing.
Class 6	Provides storage statistics for the DBM1 address space.
Class 7	Provides IBM DRDA® location statistics.
Class 8	Provides data set I/O statistics.

For the recommended collection of statistics classes 1, 3, 4, and 6, see *CICS Transaction Server for z/OS V4.2: DB2 Guide*, SC34-7164.

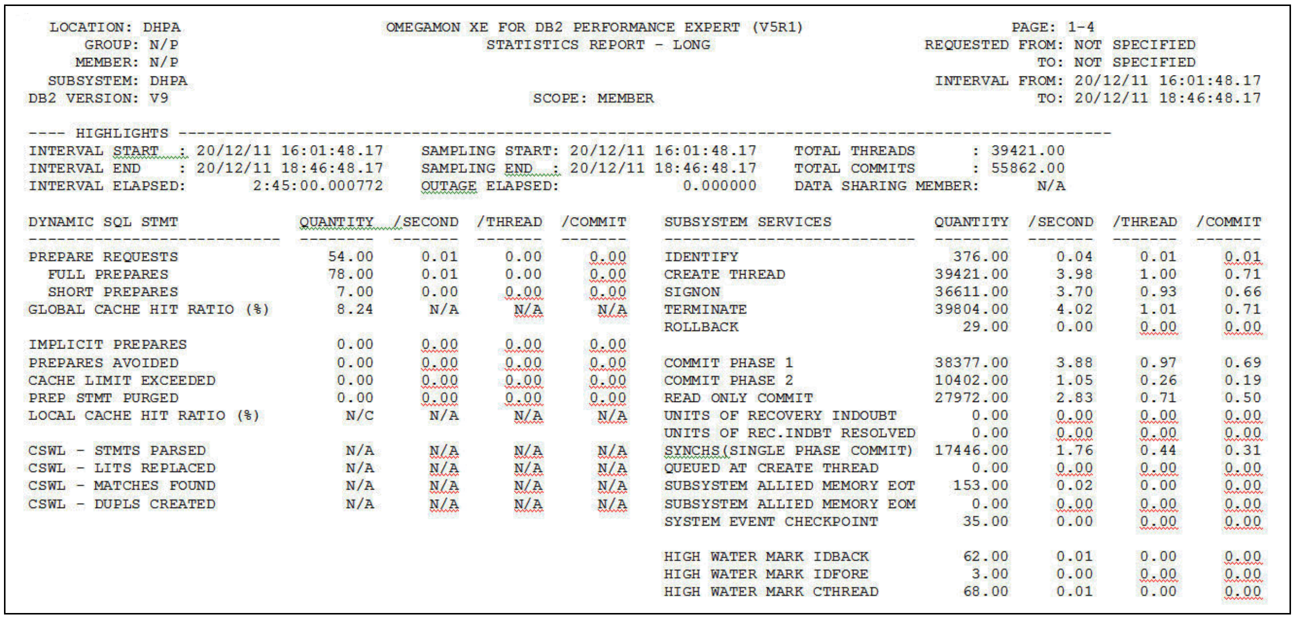
Figure 15 shows an IBM OMEGAMON DB2 Performance Expert for z/OS sample JCL that can be used to format the statistics records.



```
//S001   EXEC PGM=FPECMAIN,PARM='DATEFORMAT=MM/DD/YY',REGION=0M
//STEPLIB  DD DISP=SHR,DSN=<product>.RKANMOD
//INPUTDD  DD DISP=SHR,DSN=<smf.data>
//JOBSUMDD DD SYSOUT=*
//SYSIN    DD *
STATISTICS
    REPORT
      DSETSTAT
      LAYOUT(LONG)
EXEC
/*
```

Figure 15   DB2 Performance Expert sample JCL for statistics report

Figure 16 is an extract of the statistics highlights block. It contains a summary of all thread activity over the period. It includes the number of create-thread requests that incurred a wait due to exceeding the CTHREAD limit.



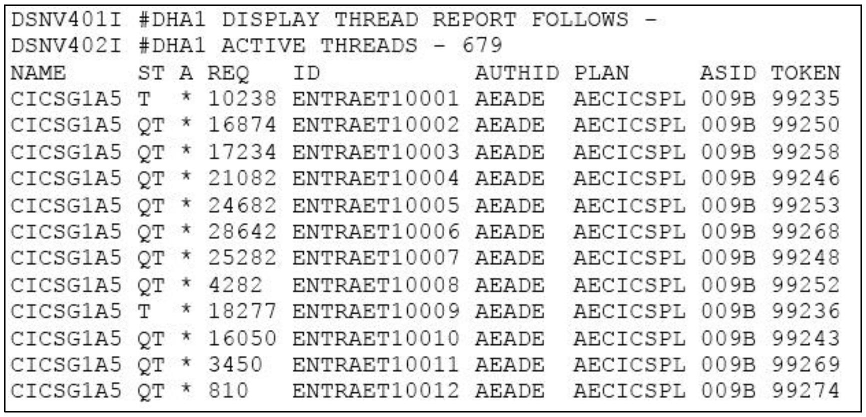
```
  LOCATION: DHPA                 OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)                 PAGE: 1-4
     GROUP: N/P                              STATISTICS REPORT - LONG                 REQUESTED FROM: NOT SPECIFIED
    MEMBER: N/P                                                                                   TO: NOT SPECIFIED
 SUBSYSTEM: DHPA                                                                       INTERVAL FROM: 20/12/11 16:01:48.17
DB2 VERSION: V9                                    SCOPE: MEMBER                                  TO: 20/12/11 18:46:48.17

---- HIGHLIGHTS ----------------------------------------------------------------------------------------------------
INTERVAL START  : 20/12/11 16:01:48.17   SAMPLING START: 20/12/11 16:01:48.17   TOTAL THREADS      : 39421.00
INTERVAL END    : 20/12/11 18:46:48.17   SAMPLING END  : 20/12/11 18:46:48.17   TOTAL COMMITS      : 55862.00
INTERVAL ELAPSED:        2:45:00.000772   OUTAGE ELAPSED:               0.000000   DATA SHARING MEMBER:       N/A

DYNAMIC SQL STMT             QUANTITY  /SECOND  /THREAD  /COMMIT   SUBSYSTEM SERVICES           QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------   ---------------------------  --------  -------  -------  -------
PREPARE REQUESTS                54.00     0.01     0.00     0.00   IDENTIFY                       376.00     0.04     0.01     0.01
  FULL PREPARES                 78.00     0.01     0.00     0.00   CREATE THREAD                39421.00     3.98     1.00     0.71
  SHORT PREPARES                 7.00     0.00     0.00     0.00   SIGNON                       36611.00     3.70     0.93     0.66
GLOBAL CACHE HIT RATIO (%)       8.24      N/A      N/A      N/A   TERMINATE                    39804.00     4.02     1.01     0.71
                                                                   ROLLBACK                        29.00     0.00     0.00     0.00
IMPLICIT PREPARES                0.00     0.00     0.00     0.00
PREPARES AVOIDED                 0.00     0.00     0.00     0.00   COMMIT PHASE 1               38377.00     3.88     0.97     0.69
CACHE LIMIT EXCEEDED             0.00     0.00     0.00     0.00   COMMIT PHASE 2               10402.00     1.05     0.26     0.19
PREP STMT PURGED                 0.00     0.00     0.00     0.00   READ ONLY COMMIT             27972.00     2.83     0.71     0.50
LOCAL CACHE HIT RATIO (%)         N/C      N/A      N/A      N/A   UNITS OF RECOVERY INDOUBT        0.00     0.00     0.00     0.00
                                                                   UNITS OF REC.INDBT RESOLVED      0.00     0.00     0.00     0.00
CSWL - STMTS PARSED               N/A      N/A      N/A      N/A   SYNCHS(SINGLE PHASE COMMIT)  17446.00     1.76     0.44     0.31
CSWL - LITS REPLACED              N/A      N/A      N/A      N/A   QUEUED AT CREATE THREAD          0.00     0.00     0.00     0.00
CSWL - MATCHES FOUND              N/A      N/A      N/A      N/A   SUBSYSTEM ALLIED MEMORY EOT    153.00     0.02     0.00     0.00
CSWL - DUPLS CREATED              N/A      N/A      N/A      N/A   SUBSYSTEM ALLIED MEMORY EOM      0.00     0.00     0.00     0.00
                                                                   SYSTEM EVENT CHECKPOINT         35.00     0.00     0.00     0.00

                                                                   HIGH WATER MARK IDBACK          62.00     0.01     0.00     0.00
                                                                   HIGH WATER MARK IDFORE           3.00     0.00     0.00     0.00
                                                                   HIGH WATER MARK CTHREAD         68.00     0.01     0.00     0.00
```

Figure 16 Extract of DB2 statistics highlights showing thread usage

For a complete explanation of the statistics blocks produced in the DB2 statistics report, see *IBM Tivoli OMEGAMON XE Performance Expert on z/OS V5.1 Report Reference*, SH12-6921.

DB2 commands

When CICS is connected to DB2, it is often necessary to display the status of the CICS threads that are connected to DB2. The DB2 **DISPLAY THREAD** command provides a detailed display of the status of all threads that are currently connected. For CICS, Figure 17 shows the output from this command.

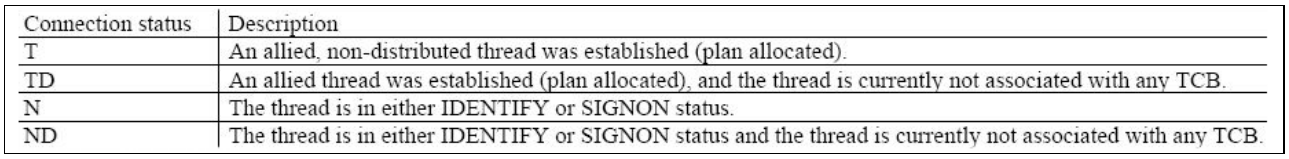


```
DSNV401I #DHA1 DISPLAY THREAD REPORT FOLLOWS -
DSNV402I #DHA1 ACTIVE THREADS - 679
NAME     ST A REQ   ID           AUTHID PLAN     ASID TOKEN
CICSG1A5 T  * 10238 ENTRAET10001 AEADE  AECICSPL 009B 99235
CICSG1A5 QT * 16874 ENTRAET10002 AEADE  AECICSPL 009B 99250
CICSG1A5 QT * 17234 ENTRAET10003 AEADE  AECICSPL 009B 99258
CICSG1A5 QT * 21082 ENTRAET10004 AEADE  AECICSPL 009B 99246
CICSG1A5 QT * 24682 ENTRAET10005 AEADE  AECICSPL 009B 99253
CICSG1A5 QT * 28642 ENTRAET10006 AEADE  AECICSPL 009B 99268
CICSG1A5 QT * 25282 ENTRAET10007 AEADE  AECICSPL 009B 99248
CICSG1A5 QT * 4282  ENTRAET10008 AEADE  AECICSPL 009B 99252
CICSG1A5 T  * 18277 ENTRAET10009 AEADE  AECICSPL 009B 99236
CICSG1A5 QT * 16050 ENTRAET10010 AEADE  AECICSPL 009B 99243
CICSG1A5 QT * 3450  ENTRAET10011 AEADE  AECICSPL 009B 99269
CICSG1A5 QT * 810   ENTRAET10012 AEADE  AECICSPL 009B 99274
```

Figure 17 Sample output from DB2 DISPLAY THREAD command

The connection status shown in the ST column contains, for most threads, the value *QT*. QT indicates that the **CREATE THREAD** request is queued, most likely due to reaching the CTHREAD limit , and the associated CICS transaction allied task is placed in a wait state.

Figure 18 contains additional connection status codes that can be seen for CICS threads that are connected to DB2.



Connection status	Description
T	An allied, non-distributed thread was established (plan allocated).
TD	An allied thread was established (plan allocated), and the thread is currently not associated with any TCB.
N	The thread is in either IDENTIFY or SIGNON status.
ND	The thread is in either IDENTIFY or SIGNON status and the thread is currently not associated with any TCB.

Figure 18 Sample subset of DB2 connection status reason codes

The connection status of *TD* represents a protected thread. *ND* represents an unprotected thread where the thread terminated but the connection retains Identify and SIGNON status so that it is ready for a thread to be created again.

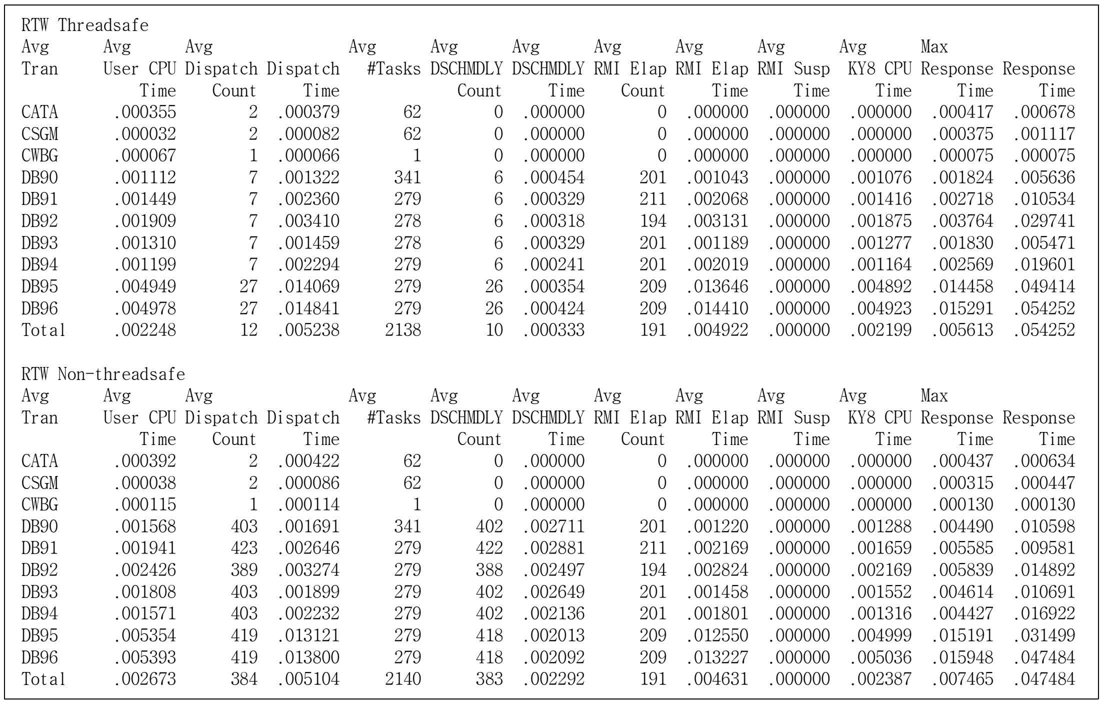
DB2 message `DSNV404I` provides a full description of the information returned by the DB2 **`DISPLAY THREAD`** command.

Analyzing applications

This section focuses on the analysis of two sample workloads. The workloads are the same application defined to CICS in different ways. Workload 1 is defined as a threadsafe workload, and workload 2 is defined as a nonthreadsafe workload. The CICS Performance Analyzer and IBM Tivoli® OMEGAMON XE for DB2 Performance Expert reports in the following sections highlight the difference between what is essentially the same application.

Summary report

Figure 19 is an extract from two CICS Performance Analyzer Summary reports, one for each workload. Notice the difference in Avg DSCHMDLY count values between the two reports. The bottom set of values are from the nonthreadsafe workload. The significant increase in task switching leads to higher CPU costs and elongated response times.

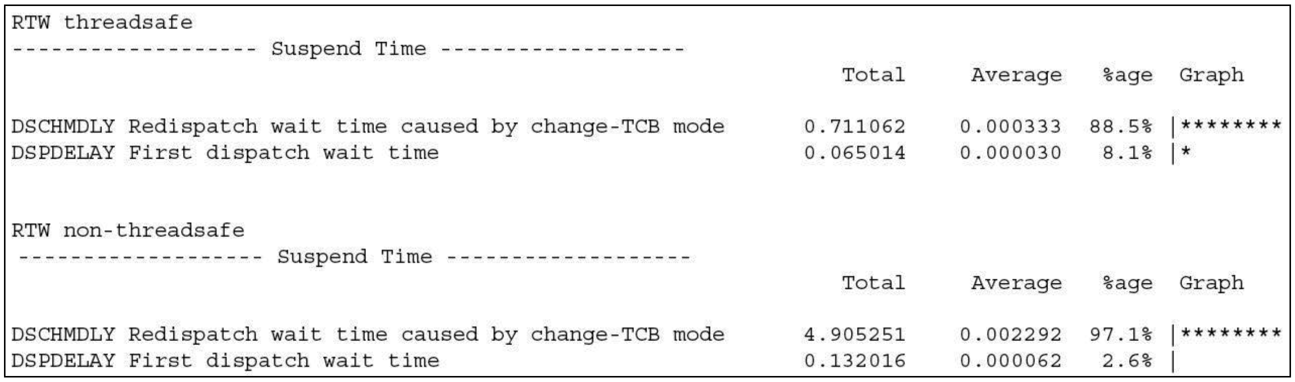


```
RTW Threadsafe
Avg      Avg      Avg               Avg      Avg      Avg      Avg      Avg      Avg      Avg      Max
Tran     User CPU Dispatch Dispatch   #Tasks DSCHMDLY DSCHMDLY RMI Elap RMI Elap RMI Susp  KY8 CPU Response Response
             Time    Count     Time             Count     Time    Count     Time     Time     Time     Time     Time
CATA      .000355        2  .000379       62        0  .000000        0  .000000  .000000  .000000  .000417  .000678
CSGM      .000032        2  .000082       62        0  .000000        0  .000000  .000000  .000000  .000375  .001117
CWBG      .000067        1  .000066        1        0  .000000        0  .000000  .000000  .000000  .000075  .000075
DB90      .001112        7  .001322      341        6  .000454      201  .001043  .000000  .001076  .001824  .005636
DB91      .001449        7  .002360      279        6  .000329      211  .002068  .000000  .001416  .002718  .010534
DB92      .001909        7  .003410      278        6  .000318      194  .003131  .000000  .001875  .003764  .029741
DB93      .001310        7  .001459      278        6  .000329      201  .001189  .000000  .001277  .001830  .005471
DB94      .001199        7  .002294      279        6  .000241      201  .002019  .000000  .001164  .002569  .019601
DB95      .004949       27  .014069      279       26  .000354      209  .013646  .000000  .004892  .014458  .049414
DB96      .004978       27  .014841      279       26  .000424      209  .014410  .000000  .004923  .015291  .054252
Total     .002248       12  .005238     2138       10  .000333      191  .004922  .000000  .002199  .005613  .054252

RTW Non-threadsafe
Avg      Avg      Avg               Avg      Avg      Avg      Avg      Avg      Avg      Avg      Max
Tran     User CPU Dispatch Dispatch   #Tasks DSCHMDLY DSCHMDLY RMI Elap RMI Elap RMI Susp  KY8 CPU Response Response
             Time    Count     Time             Count     Time    Count     Time     Time     Time     Time     Time
CATA      .000392        2  .000422       62        0  .000000        0  .000000  .000000  .000000  .000437  .000634
CSGM      .000038        2  .000086       62        0  .000000        0  .000000  .000000  .000000  .000315  .000447
CWBG      .000115        1  .000114        1        0  .000000        0  .000000  .000000  .000000  .000130  .000130
DB90      .001568      403  .001691      341      402  .002711      201  .001220  .000000  .001288  .004490  .010598
DB91      .001941      423  .002646      279      422  .002881      211  .002169  .000000  .001659  .005585  .009581
DB92      .002426      389  .003274      279      388  .002497      194  .002824  .000000  .002169  .005839  .014892
DB93      .001808      403  .001899      279      402  .002649      201  .001458  .000000  .001552  .004614  .010691
DB94      .001571      403  .002232      279      402  .002136      201  .001801  .000000  .001316  .004427  .016922
DB95      .005354      419  .013121      279      418  .002013      209  .012550  .000000  .004999  .015191  .031499
DB96      .005393      419  .013800      279      418  .002092      209  .013227  .000000  .005036  .015948  .047484
Total     .002673      384  .005104     2140      383  .002292      191  .004631  .000000  .002387  .007465  .047484
```

Figure 19 Extract from the CICS Performance Analyzer Summary report for both workloads

Recap report

The delays caused by the TCB switching are shown in the CICS Performance Analyzer WAITANALYSIS report. Figure 20 includes snippets of the Wait Analysis Recap report. Although both workloads incurred delays due to change mode TCB activity, the average delay for the nonthreadsafe workload (0.002292) far exceeds the delay for the threadsafe workload (0.000333).



```
RTW threadsafe
------------------- Suspend Time -------------------
                                                         Total     Average   %age  Graph

DSCHMDLY Redispatch wait time caused by change-TCB mode   0.711062   0.000333  88.5% |********
DSPDELAY First dispatch wait time                         0.065014   0.000030   8.1% |*


RTW non-threadsafe
 ------------------- Suspend Time -------------------
                                                         Total     Average   %age  Graph

DSCHMDLY Redispatch wait time caused by change-TCB mode   4.905251   0.002292  97.1% |********
DSPDELAY First dispatch wait time                         0.132016   0.000062   2.6% |
```

Figure 20 CICS Performance Analyzer WAITANALYSIS Report: Recap report for two workloads

The CICS Performance Analyzer DB2 report merges the CICS and DB2 SMF reports to provide an overview of activity in CICS and DB2.

Short report

The report in Figure 21 contains information from the DB2 SMF accounting records. Notice that the average transaction times are the same, but the CPU time from the CICS Performance Analyzer summary report is broken down into User, Thread, and In-DB2 categories. For a description of the information in the report, see *CICS Performance Analyzer Report Reference,* SC34-7154.

```
RTW threadsafe

Tran/ Program/ #Tasks/  ...........Average Elapsed Time...........  .....Average CPU Time.....  .....Average Count....... #Abends
SSID  Planname #Threads Response  Thread   In-DB2  DB2ConWt DB2ThdWt    User    Thread   In-DB2   DB2Reqs  GetPage SysPgUpd

DB90  DB900001      341  .001824                   .000000  .000000   .001112                      200.0                           0
DI2A  DB9D          341           .366544  .000723                              .001016  .000640              47.2       .0

DB91  DB910001      279  .002717                   .000000  .000000   .001448                      210.0                           0
DI2A  DB9D          279           .222263  .001796                              .001358  .000942             106.3     40.0

DB92  DB920001      278  .003763                   .000000  .000000   .001909                      193.0                           0
DI2A  DB9D          278           .223014  .002859                              .001817  .001417             148.1    176.3

DB93  DB930001      278  .001830                   .000000  .000000   .001309                      200.0                           0
DI2A  DB9D          278           .223014  .000895                              .001223  .000831               3.4       .0

DB94  DB940001      279  .002569                   .000000  .000000   .001198                      200.0                           0
DI2A  DB9D          279           .222229  .001753                              .001107  .000728              40.1       .0

DB95  DB950001      279  .014458                   .000000  .000000   .004948                      198.0                           0
DI2A  DB9D         3069           .034215  .001192                              .000419  .000368              43.2     14.0

DB96  DB960001      279  .015291                   .000000  .000000   .004977                      198.0                           0
DI2A  DB9D         3068           .034520  .001261                              .000422  .000371              43.3     14.0


RTW non-threadsafe

Tran/ Program/ #Tasks/  ...........Average Elapsed Time...........  .....Average CPU Time.....  .....Average Count....... #Abends
SSID  Planname #Threads Response  Thread   In-DB2  DB2ConWt DB2ThdWt    User    Thread   In-DB2   DB2Reqs  GetPage SysPgUpd

DB90  DB900001      341  .004489                   .000000  .000000   .001567                      200.0                           0
DI2A  DB9D          341           .360671  .000909                              .001493  .000854              46.4       .0

DB91  DB910001      279  .005585                   .000000  .000000   .001941                      210.0                           0
DI2A  DB9D          279           .222199  .001854                              .001881  .001197             106.6     40.0

DB92  DB920001      279  .005838                   .000000  .000000   .002425                      193.1                           0
DI2A  DB9D          279           .222200  .002529                              .002380  .001739             146.3    174.5

DB93  DB930001      279  .004613                   .000000  .000000   .001808                      200.0                           0
DI2A  DB9D          279           .222199  .001149                              .001766  .001109               2.0       .0

DB94  DB940001      279  .004427                   .000000  .000000   .001571                      200.0                           0
DI2A  DB9D          279           .222197  .001513                              .001525  .000907              42.5       .0

DB95  DB950001      279  .015190                   .000000  .000000   .005354                      198.0                           0
DI2A  DB9D         3069           .036854  .001090                              .000459  .000391              43.2     14.0

DB96  DB960001      279  .015947                   .000000  .000000   .005392                      198.0                           0
DI2A  DB9D         3069           .040067  .001151                              .000462  .000394              43.2     14.0
```

Figure 21 Extract from the CICS Performance Analyzer DB2 SHORT(SUMMARY) report

Expert accounting report

The Tivoli OMEGAMON XE Performance Expert on z/OS 5.1 provides a report that is derived from the DB2 accounting records and that shows detailed information at the plan level. Figure 22 shows the sample JCL used to create an accounting report from the data collected for workload 1.

```
//S001   EXEC PGM=FPECMAIN,PARM='DATEFORMAT=MM/DD/YY',
//          REGION=0M
//STEPLIB  DD DISP=SHR,DSN=DSN=<library>.RKANMOD
//INPUTDD  DD DISP=SHR,DSN=SYS1.MVA1.MAND
//JOBSUMDD DD SYSOUT=*
//SYSIN    DD *
ACCOUNTING
   REPORT SCOPE(MEMBER)
     LAYOUT(LONG)
EXEC
/*
```

Figure 22   Sample JCL to create the DB2 Performance Expert accounting report

The report is large and, therefore, is not suitable for inclusion in its entirety. Figure 23 contains an extract from the report that shows a breakdown of the elapse, CPU, and suspend times for the plan used by CICS in workload 1.

```
PRIMAUTH: CLARKET  PLANNAME: DB9D

ELAPSED TIME DISTRIBUTION                                           CLASS 2 TIME DISTRIBUTION
---------------------------------------------------------------     -------------------------------------------------------------
APPL   |================================================> 98%       CPU    |==================> 36%
DB2    |> 1%                                                        SECPU  |
SUSP   |> 1%                                                        NOTACC |> 1%
                                                                    SUSP   |===============================> 63%

AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT   HIGHLIGHTS
------------  ----------  ----------  ----------    --------------------  ------------  --------   --------------------------
ELAPSED TIME    0.087227    0.001309         N/P    LOCK/LATCH(DB2+IRLM)      0.000036      0.16   #OCCURRENCES    :     8228
 NONNESTED      0.087227    0.001309         N/A     IRLM LOCK+LATCH          0.000035      0.11   #ALLIEDS        :     8228
 STORED PROC    0.000000    0.000000         N/A     DB2 LATCH                0.000000      0.04   #ALLIEDS DISTRIB:        0
 UDF            0.000000    0.000000         N/A    SYNCHRON. I/O             0.000239      0.19   #DBATS          :        0
 TRIGGER        0.000000    0.000000         N/A     DATABASE I/O             0.000237      0.19   #DBATS DISTRIB. :        0
                                                     LOG WRITE I/O            0.000002      0.00   #NO PROGRAM DATA:     8228
CP CPU TIME     0.000653    0.000471         N/P    OTHER READ I/O            0.000019      0.01   #NORMAL TERMINAT:     8228
 AGENT          0.000653    0.000471         N/A    OTHER WRTE I/O            0.000000      0.00   #DDFRRSAF ROLLUP:        0
  NONNESTED     0.000653    0.000471         N/P    SER.TASK SWTCH            0.000526      0.87   #ABNORMAL TERMIN:        0
  STORED PRC    0.000000    0.000000         N/A     UPDATE COMMIT            0.000525      0.87   #CP/X PARALLEL. :        0
  UDF           0.000000    0.000000         N/A     OPEN/CLOSE               0.000000      0.00   #IO PARALLELISM :        0
  TRIGGER       0.000000    0.000000         N/A     SYSLGRNG REC             0.000000      0.00   #INCREMENT. BIND:        0
 PAR.TASKS      0.000000    0.000000         N/A     EXT/DEL/DEF              0.000000      0.00   #COMMITS        :     8223
                                                     OTHER SERVICE            0.000001      0.00   #ROLLBACKS      :        0
 SECP CPU       0.000000         N/A         N/A    ARC.LOG(QUIES)            0.000000      0.00   #SVPT REQUESTS  :        0
                                                    LOG READ                  0.000000      0.00   #SVPT RELEASE   :        0
SE CPU TIME     0.000000    0.000000         N/A    DRAIN LOCK                0.000000      0.00   #SVPT ROLLBACK  :        0
 NONNESTED      0.000000    0.000000         N/A    CLAIM RELEASE             0.000000      0.00   MAX SQL CASC LVL:        0
 STORED PROC    0.000000    0.000000         N/A    PAGE LATCH                0.000000      0.00   UPDATE/COMMIT   :     9.78
 UDF            0.000000    0.000000         N/A    NOTIFY MSGS               0.000000      0.00   SYNCH I/O AVG.  : 0.001255
 TRIGGER        0.000000    0.000000         N/A    GLOBAL CONTENTION         0.000000      0.00
                                                    COMMIT PH1 WRITE I/O      0.000000      0.00
 PAR.TASKS      0.000000    0.000000         N/A    ASYNCH CF REQUESTS        0.000000      0.00
                                                    TCP/IP LOB XML            0.000000      0.00
SUSPEND TIME    0.000000    0.000820         N/A    TOTAL CLASS 3             0.000820      1.24
```

Figure 23   Extract from the DB2 Performance Expert accounting report with the elapse and CPU breakdown

End-to-end analysis

Frequently, it is necessary to understand the end-to-end processing of a CICS transaction. This section provides sample JCL and report extracts from CICS Performance Analyzer and DB2 Performance Expert, which are used to analyze a single transaction. The transaction was selected at random from the CICS Performance Analyzer List report.

Figure 24 shows the JCL that was used to run all CICS Performance Analyzer reports.

```
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD DISP=SHR,DSN=<hlq>.SCPALINK
//SYSPRINT DD SYSOUT=*
//* SMF Input Files
//SMFIN001 DD DISP=SHR,DSN=SYS1.MVA1.MAND
//CPAXW001 DD DSN=&&CPAXW001,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//CPASWK01 DD DSN=&&CPASWK01,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//CPASWK02 DD DSN=&&CPASWK02,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//CPASWK03 DD DSN=&&CPASWK03,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//CPASWK04 DD DSN=&&CPASWK04,DISP=(NEW,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSOUT   DD SYSOUT=*
//SYSIN DD *
```

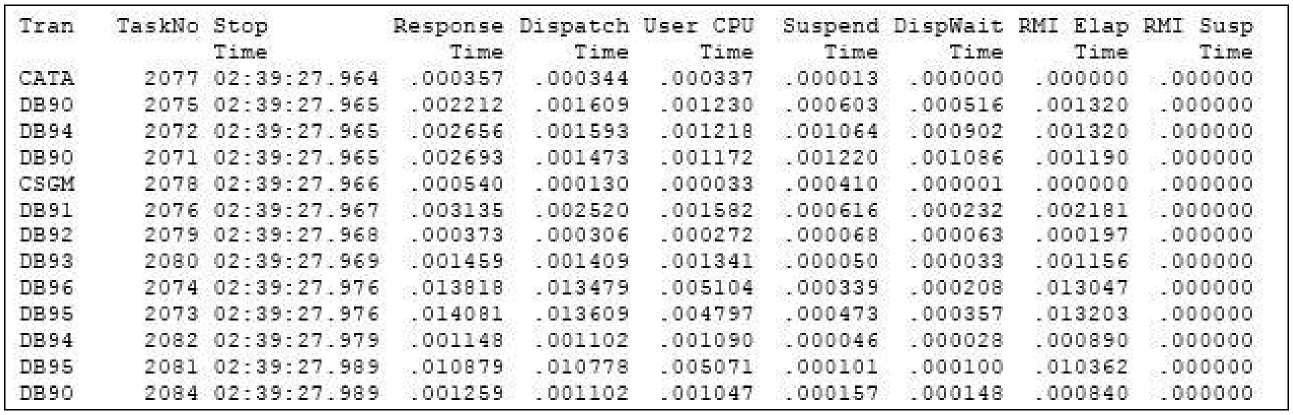
Figure 24   CICS Performance Analyzer JCL for the sample reports

Figure 25 shows the CICS Performance Analyzer command syntax that is used to generate the LIST report.

```
CICSPA IN(SMFIN001),
    PRECISION(6),
    LINECNT(60),
    FORMAT(':','/'),
    LIST(OUTPUT(REPORT1),
      SELECT(PERFORMANCE(
      INC(,
       START(FROM(,00:00:00.00),
            TO(,23:59:00.00)),
        STOP(FROM(,00:00:00.00),
            TO(,23:59:00.00))))),
      FIELDS(TRAN,
          TASKNO,
          STOP(TIMET),
          RESPONSE,
          DISPATCH(TIME),
          CPU(TIME),
          SUSPEND(TIME),
          DISPWAIT(TIME),
          RMITIME,
          RMISUSP),
      TITLE1('LIST REPORT - WORKLOAD 1'))
```

Figure 25   CICS Performance Analyzer command used to generate the LIST report

Figure 26 contains an extract from the generated report. This example uses the task number 2074, which has a transaction identifier of DB96.



```
Tran   TaskNo Stop         Response Dispatch User CPU  Suspend DispWait RMI Elap RMI Susp
              Time             Time     Time     Time     Time     Time     Time     Time
CATA     2077 02:39:27.964  .000357  .000344  .000337  .000013  .000000  .000000  .000000
DB90     2075 02:39:27.965  .002212  .001609  .001230  .000603  .000516  .001320  .000000
DB94     2072 02:39:27.965  .002656  .001593  .001218  .001064  .000902  .001320  .000000
DB90     2071 02:39:27.965  .002693  .001473  .001172  .001220  .001086  .001190  .000000
CSGM     2078 02:39:27.966  .000540  .000130  .000033  .000410  .000001  .000000  .000000
DB91     2076 02:39:27.967  .003135  .002520  .001582  .000616  .000232  .002181  .000000
DB92     2079 02:39:27.968  .000373  .000306  .000272  .000068  .000063  .000197  .000000
DB93     2080 02:39:27.969  .001459  .001409  .001341  .000050  .000033  .001156  .000000
DB96     2074 02:39:27.976  .013818  .013479  .005104  .000339  .000208  .013047  .000000
DB95     2073 02:39:27.976  .014081  .013609  .004797  .000473  .000357  .013203  .000000
DB94     2082 02:39:27.979  .001148  .001102  .001090  .000046  .000028  .000890  .000000
DB95     2081 02:39:27.989  .010879  .010778  .005071  .000101  .000100  .010362  .000000
DB90     2084 02:39:27.989  .001259  .001102  .001047  .000157  .000148  .000840  .000000
```

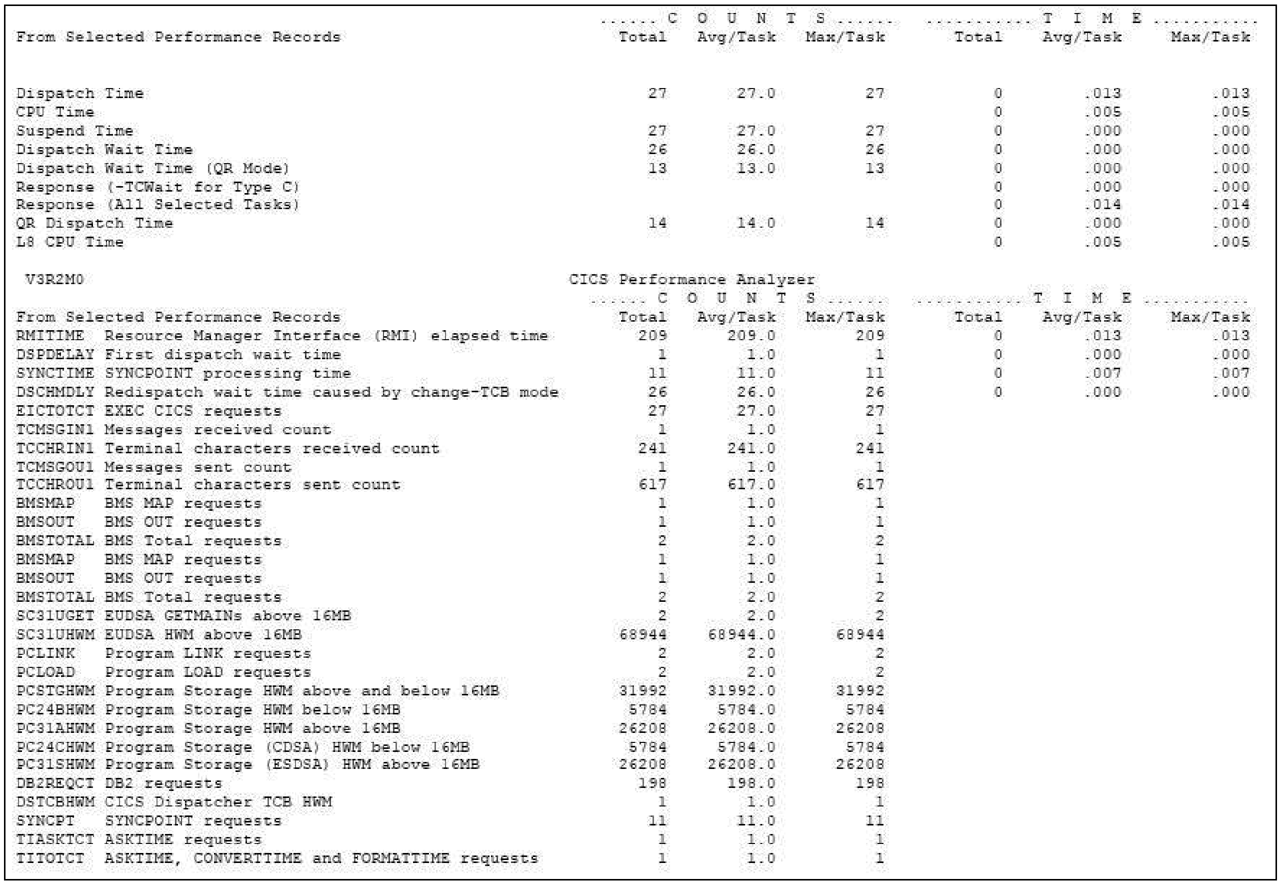
Figure 26 CICS Performance Analyzer output from LIST report

Numerous CICS Performance Analyzer reports are available to provide information about the analysis of transaction execution, but the TOTAL report gives the most detailed breakdown of the processing that is done (Figure 27).

```
CICSPA IN(SMFIN001),
       APPLID(CICSG1A5),
       LINECNT(60),
       FORMAT(':','/'),
       PRECISION(4),
       TOTAL(OUTPUT(REPORT1),
            TITLE1('PA TOTAL REPORT – DB96 - 2074'),
            SELECT(PERFORMANCE(
            INC(TRAN(DB96),TASKNO(2074),
               START(FROM(,00:00:00.00),
                     TO(,23:59:00.00)),
                STOP(FROM(,00:00:00.00),
                     TO(,23:59:00.00))))))
```

Figure 27 CICS Performance Analyzer command used to generate the TOTAL report

The output generated by the TOTAL report is considerable. Figure 28 is a condensed sample of the output produced for the transaction. Of interest is the CPU time that is attributed to the QR and L8 TCBs, the DB2 request count (198), and sync point count (11).



```
                                                          ...... C  O  U  N  T  S ......    ........... T  I  M  E ...........
From Selected Performance Records                            Total    Avg/Task   Max/Task       Total    Avg/Task     Max/Task


Dispatch Time                                                   27        27.0         27           0        .013         .013
CPU Time                                                                                            0        .005         .005
Suspend Time                                                    27        27.0         27           0        .000         .000
Dispatch Wait Time                                              26        26.0         26           0        .000         .000
Dispatch Wait Time (QR Mode)                                    13        13.0         13           0        .000         .000
Response (-TCWait for Type C)                                                                       0        .000         .000
Response (All Selected Tasks)                                                                       0        .014         .014
QR Dispatch Time                                                14        14.0         14           0        .000         .000
L8 CPU Time                                                                                         0        .005         .005

 V3R2M0                                                 CICS Performance Analyzer
                                                          ...... C  O  U  N  T  S ......   ........... T  I  M  E ...........
From Selected Performance Records                            Total    Avg/Task   Max/Task       Total    Avg/Task     Max/Task
RMITIME  Resource Manager Interface (RMI) elapsed time         209       209.0        209           0        .013         .013
DSPDELAY First dispatch wait time                                1         1.0          1           0        .000         .000
SYNCTIME SYNCPOINT processing time                              11        11.0         11           0        .007         .007
DSCHMDLY Redispatch wait time caused by change-TCB mode         26        26.0         26           0        .000         .000
EICTOTCT EXEC CICS requests                                     27        27.0         27
TCMSGIN1 Messages received count                                 1         1.0          1
TCCHRIN1 Terminal characters received count                    241       241.0        241
TCMSGOU1 Messages sent count                                     1         1.0          1
TCCHROU1 Terminal characters sent count                        617       617.0        617
BMSMAP   BMS MAP requests                                        1         1.0          1
BMSOUT   BMS OUT requests                                        1         1.0          1
BMSTOTAL BMS Total requests                                      2         2.0          2
BMSMAP   BMS MAP requests                                        1         1.0          1
BMSOUT   BMS OUT requests                                        1         1.0          1
BMSTOTAL BMS Total requests                                      2         2.0          2
SC31UGET EUDSA GETMAINs above 16MB                               2         2.0          2
SC31UHWM EUDSA HWM above 16MB                                68944     68944.0      68944
PCLINK   Program LINK requests                                   2         2.0          2
PCLOAD   Program LOAD requests                                   2         2.0          2
PCSTGHWM Program Storage HWM above and below 16MB            31992     31992.0      31992
PC24BHWM Program Storage HWM below 16MB                       5784      5784.0       5784
PC31AHWM Program Storage HWM above 16MB                      26208     26208.0      26208
PC24CHWM Program Storage (CDSA) HWM below 16MB                5784      5784.0       5784
PC31SHWM Program Storage (ESDSA) HWM above 16MB              26208     26208.0      26208
DB2REQCT DB2 requests                                          198       198.0        198
DSTCBHWM CICS Dispatcher TCB HWM                                 1         1.0          1
SYNCPT   SYNCPOINT requests                                     11        11.0         11
TIASKTCT ASKTIME requests                                        1         1.0          1
TITOTCT  ASKTIME, CONVERTTIME and FORMATTIME requests            1         1.0          1
```

Figure 28 CICS Performance Analyzer output from TOTAL report

The CICS Performance Analyzer DB2 report consolidates the CICS and DB2 SMF and provides a more detailed breakdown of the processing that is performed in DB2. Figure 29 shows the CICS Performance Analyzer command syntax to generate the report. Several versions of the DB2 report provide short and long summaries. This example uses the LIST option with a filter for this specific transaction.

```
CICSPA IN(SMFIN001),
     LINECNT(60),
     FORMAT(':','/'),
     DB2(LIST(ALL),
        TITLE('RTW threadsafe'),
        OUTPUT(REPORT1),
        SELECT(PERFORMANCE(,
        INC(TRAN(DB96),TASKNO(2074),
        START(FROM(,00:00:00.00),
        TO(,23:59:59.00)))))
```

Figure 29 CICS Performance Analyzer command used to generate the DB2 report

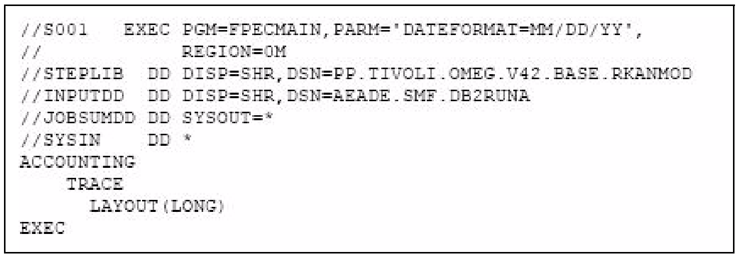
Figure 30 is an extract from the report shown in Figure 29 on page 32. Because the output can be considerable, only an extract is provided. The report shows a summary line followed by three distinct blocks of output. Each block of output is for transaction DB96, task number 2074. In total, 11 blocks of output are produced, one for each sync point that is performed by the transaction (as reported in the TOTAL report) and as shown in the report under the UOW Seq column. Because the transaction is terminal attached, you expect the threads to be released at sync point. In this example, because the threads are protected, the ENTRDB960007 thread identifier remains for the duration of the transaction.

To correlate this information with the DB2 Performance Expert trace reports, notice the UOWID for the transaction, which in this case is D148F8FE7A47.

```
Tran/ Userid/ Program/               UOW R              ..DB2 Wait Time..   DB2    User CPU                             Response A
SSID  Authid  Planname  APPLID   Task Seq T Term  LUName   Connect  Thread   ReqCnt    Time      Start Time     Stop Time     Time   B

DB96 CLARKET  DB960001 IYCUZC19   2074  11 T 0099 TC0099    .000000  .000000      198  .005103   2:39:27.962  2:39:27.976  .013818

DI2A CLARKET  DB9D     IYCUZC19   2074  Thread Identification   ID=ENTRDB960007  NETName=GBIBMIYA.TC0099    UOWID=D148F8FE7A47
                                        Begin Time:  3:39:27.963303 12/14/11  End Time:  3:39:27.966413 12/14/11
                                        Class1: Thread Time     Elapsed= .003110  CPU= .000918
                                        Class2: In-DB2 Time     Elapsed= .003027  CPU= .000852
                                        Class3: Suspend Time    Total  = .002128  I/O= .001199  Lock/Latch= .000000  Other= .000929
                                        Buffer Manager Summary  GtPgRq=      43  SyPgUp=      14
                                        Locking Summary         Suspnd=       0  DeadLk=       0  TmeOut=       0  MxPgLk=       3
                                        SQL DML Query/Update    Sel=       5  Ins=       3  Upd=       2  Del=       2
                                        SQL DML 'Other'         Des=       0  Pre=       0  Ope=       2  Fet=       2  Clo=       2

DI2A CLARKET  DB9D     IYCUZC19   2074  Thread Identification   ID=ENTRDB960007  NETName=GBIBMIYA.TC0099    UOWID=D148F8FE7A47
                                        Begin Time:  3:39:27.966443 12/14/11  End Time:  3:39:27.967665 12/14/11
                                        Class1: Thread Time     Elapsed= .001222  CPU= .000465
                                        Class2: In-DB2 Time     Elapsed= .001153  CPU= .000410
                                        Class3: Suspend Time    Total  = .000737  I/O= .000000  Lock/Latch= .000046  Other= .000691
                                        Buffer Manager Summary  GtPgRq=      43  SyPgUp=      14
                                        Locking Summary         Suspnd=       0  DeadLk=       0  TmeOut=       0  MxPgLk=       3
                                        SQL DML Query/Update    Sel=       5  Ins=       3  Upd=       2  Del=       2
                                        SQL DML 'Other'         Des=       0  Pre=       0  Ope=       2  Fet=       2  Clo=       2

DI2A CLARKET  DB9D     IYCUZC19   2074  Thread Identification   ID=ENTRDB960007  NETName=GBIBMIYA.TC0099    UOWID=D148F8FE7A47
                                        Begin Time:  3:39:27.967684 12/14/11  End Time:  3:39:27.968516 12/14/11
                                        Class1: Thread Time     Elapsed= .000832  CPU= .000373
                                        Class2: In-DB2 Time     Elapsed= .000763  CPU= .000326
                                        Class3: Suspend Time    Total  = .000436  I/O= .000000  Lock/Latch= .000000  Other= .000436
                                        Buffer Manager Summary  GtPgRq=      43  SyPgUp=      14
                                        Locking Summary         Suspnd=       0  DeadLk=       0  TmeOut=       0  MxPgLk=       3
                                        SQL DML Query/Update    Sel=       5  Ins=       3  Upd=       2  Del=       2
                                        SQL DML 'Other'         Des=       0  Pre=       0  Ope=       2  Fet=       2  Clo=       2
```

Figure 30 Extract from the CICS Performance Analyzer DB2 report with LIST options

You use the DB2 Performance Expert product to format the DB2 accounting records. Figure 31 shows the JCL and command syntax to generate the report.



```
//S001    EXEC PGM=FPECMAIN,PARM='DATEFORMAT=MM/DD/YY',
//             REGION=0M
//STEPLIB  DD DISP=SHR,DSN=PP.TIVOLI.OMEG.V42.BASE.RKANMOD
//INPUTDD  DD DISP=SHR,DSN=AEADE.SMF.DB2RUNA
//JOBSUMDD DD SYSOUT=*
//SYSIN    DD *
ACCOUNTING
    TRACE
      LAYOUT(LONG)
EXEC
```

Figure 31 JCL and DB2 Performance Expert command syntax to generate the accounting trace report

The output from this command is considerable. Similar to the CICS Performance Analyzer report, it has 11 entries for each sync point that is performed.

Figure 32 is the first page of the first entry. Notice that the CICS INS field contains the UOWID of D148F8FE7A47 from the CICS Performance Analyzer DB2 report. The accounting trace in its entirety provides a detailed breakdown of the activity performed in DB2 and the resources that are used during its undertaking.

```
END TIME    : 12/14/11 02:39:27.96    PROD VER: N/P            LUW NET: GBIBMIYA       CICS INS: D148F8FE7A47
REQUESTER   : DSNV10P3                CORRNAME: DB96           LUW LUN: IYCUZDB0
MAINPACK    : DB9D                    CORRNMBR: ENTR           LUW INS: C8D148EBAE58    ENDUSER : 'BLANK'
PRIMAUTH    : CLARKET                 CONNTYPE: CICS           LUW SEQ:          498    TRANSACT: 'BLANK'
ORIGAUTH    : CLARKET                 CONNECT : IYCUZC19                                WSNAME  : 'BLANK'

MVS ACCOUNTING DATA    : 1
ACCOUNTING TOKEN(CHAR): N/A
ACCOUNTING TOKEN(HEX) : N/A

ELAPSED TIME DISTRIBUTION                                          CLASS 2 TIME DISTRIBUTION
----------------------------------------------------------------   ----------------------------------------------------------------
APPL   |=> 3%                                                      CPU    |==============> 28%
DB2    |==============> 29%                                        SECPU  |
SUSP   |==================================> 68%                     NOTACC |> 1%
                                                                   SUSP   |===================================> 70%

   LOCATION: DSNV10P3               OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)                PAGE: 1-997
      GROUP: DSNV10P3                          ACCOUNTING TRACE - LONG                 REQUESTED FROM: NOT SPECIFIED
     MEMBER: DI2A                                                                                TO: NOT SPECIFIED
  SUBSYSTEM: DI2A                                                                       ACTUAL FROM: 12/14/11 02:39:25.96
DB2 VERSION: V10

---- IDENTIFICATION --------------------------------------------------------------------------------------------------------------
ACCT TSTAMP: 12/14/11 02:39:27.96    PLANNAME: DB9D           WLM SCL: 'BLANK'        CICS NET: GBIBMIYA
BEGIN TIME : 12/14/11 02:39:27.96    PROD TYP: N/P                                    CICS LUN: TC0099
END TIME   : 12/14/11 02:39:27.96    PROD VER: N/P            LUW NET: GBIBMIYA       CICS INS: D148F8FE7A47
REQUESTER  : DSNV10P3                CORRNAME: DB96           LUW LUN: IYCUZDB0
MAINPACK   : DB9D                    CORRNMBR: ENTR           LUW INS: C8D148EBAE58    ENDUSER : 'BLANK'
PRIMAUTH   : CLARKET                 CONNTYPE: CICS           LUW SEQ:          498    TRANSACT: 'BLANK'
ORIGAUTH   : CLARKET                 CONNECT : IYCUZC19                                WSNAME  : 'BLANK'

TIMES/EVENTS  APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   ELAPSED TIME    EVENTS     HIGHLIGHTS
------------  ----------  ----------  ----------    --------------------  ------------  --------     --------------------------
ELAPSED TIME    0.003111    0.003027         N/P    LOCK/LATCH(DB2+IRLM)      0.000000         0     THREAD TYPE   : ALLIED
 NONNESTED      0.003111    0.003027         N/A     IRLM LOCK+LATCH          0.000000         0     TERM.CONDITION: NORMAL
 STORED PROC    0.000000    0.000000         N/A     DB2 LATCH                0.000000         0     INVOKE REASON : RESIGNON
 UDF            0.000000    0.000000         N/A    SYNCHRON. I/O             0.001200         3     PARALLELISM   : NO
 TRIGGER        0.000000    0.000000         N/A     DATABASE I/O             0.001200         3     QUANTITY      :          0
                                                     LOG WRITE I/O            0.000000         0     COMMITS       :          1
CP CPU TIME     0.000918    0.000853         N/P    OTHER READ I/O            0.000000         0     ROLLBACK      :          0
 AGENT          0.000918    0.000853         N/A    OTHER WRTE I/O            0.000000         0     SVPT REQUESTS :          0
  NONNESTED     0.000918    0.000853         N/P    SER.TASK SWTCH            0.000930         1     SVPT RELEASE  :          0
  STORED PRC    0.000000    0.000000         N/A     UPDATE COMMIT            0.000930         1     SVPT ROLLBACK :          0
  UDF           0.000000    0.000000         N/A     OPEN/CLOSE               0.000000         0     INCREM.BINDS  :          0
  TRIGGER       0.000000    0.000000         N/A     SYSLGRNG REC             0.000000         0     UPDATE/COMMIT :       7.00
 PAR.TASKS      0.000000    0.000000         N/A     EXT/DEL/DEF              0.000000         0     SYNCH I/O AVG.:   0.000400
```

Figure 32 Extract from the DB2PE ACCOUNTREC command with the trace option

The team who wrote this paper

This paper was produced by a team of specialists from around the world who is working at the International Technical Support Organization (ITSO) in Hursley, England.

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

Trevor Clarke is a CICS Performance Specialist at the IBM Hursley laboratory in the UK. He has over 30 years of experience in the IT industry, working for IBM and as a customer. For most of this time, he worked with CICS in technical support, application development, performance, and education roles. His areas of expertise include DB2, messaging, and networking. Trevor holds a Bachelor of Science degree in Mathematics from Southampton University.

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

Now you can become a published author, too!

Stay connected to IBM Redbooks publications

Notices

This document REDP-4860-00 was created or updated on July 31, 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.

Redpaper™

Trademarks