Threadsafe Conversation Techniques for CICS Applications

Noel Javier C. Sales

Software Engineer zCICS Development

IBM CICS® User Conference 2009

© 2009 IBM Corporation

Notes

- **IBM CICS customers who convert their IBM DB2® or IBM WebSphere® MQ applications to threadsafe can achieve significant cpu savings and improved throughput. This session describes how to identify non-threadsafe activity using tools like IBM CICS Interdependency Analyzer and convert those programs to meet threadsafe requirements. We will also discuss how to identify programs that have the maximum potential for CPU savings, and why some programs do not meet their expected performance improvements. We will also describe how to tell if programs are maximizing their CPU savings using tools like IBM CICS Performance Analyzer and what to do with the programs that don't. We will include a brief overview of issues related to the use of OPENAPI programs, Task Related User Exits, type of files used and storage keys.**

© 2009 IBM Corporation

Notes

- **IBM CICS customers who convert their IBM DB2® or IBM WebSphere® MQ applications to threadsafe can achieve significant cpu savings and improved throughput. This session describes how to identify non-threadsafe activity using tools like IBM CICS Interdependency Analyzer and convert those programs to meet threadsafe requirements. We will also discuss how to identify programs that have the maximum potential for CPU savings, and why some programs do not meet their expected performance improvements. We will also describe how to tell if programs are maximizing their CPU savings using tools like IBM CICS Performance Analyzer and what to do with the programs that don't. We will include a brief overview of issues related to the use of OPENAPI programs, Task Related User Exits, type of files used and storage keys.**

© 2009 IBM Corporation

This slide left intentionally blank.

© 2009 IBM Corporation

Terminology

- Quasi-reentrant (QR) TCB
  - The main CICS TCB under which all application code runs prior to OTE
  - CICS dispatcher subdispatches work, so each CICS task has a slice of the action
  - A CICS task gives up control via a CICS dispatcher wait
  - Only one CICS user task is active at any one time
- Quasi-reentrant programs
  - Same program can be invoked by more than one CICS task
    - But only one CICS task is active at any one time
  - Quasi-reentrancy allows programs to share virtual storage e.g. CWA without the need to protect against concurrent update
  - CICS code takes advantage of quasi-reentrancy, e.g. can avoid locking if code always runs on QR.

© 2009 IBM Corporation

Notes

- **Prior to OTE, all application code runs under the main CICS TCB called the quasi-reentrant (QR) TCB. The CICS dispatcher subdispatches use of the TCB between the CICS tasks. Each task voluntarily gives up control when it issues a CICS service that issues a CICS dispatcher wait. There is only ever one CICS task active at any one time on the QR TCB.**
- **Programs are said to be quasi-reentrant programs because they take advantage of the behaviour of the CICS dispatcher and the QR TCB, in particular that there is only ever one CICS task active under the QR TCB. This means that although the same program can be being executed by multiple CICS tasks, only one of those CICS tasks is active at any one point in time. Contrast this with a situation whereby multiple instances of the same program are executing each under a separate TCB. In this situation multiple tasks would be active in the same program at the same time and the program would have to be fully MVS reentrant.**
- **Quasi reentrant programs can access shared resources such as the common work area (CWA) or shared storage obtained via EXEC CICS GETMAIN SHARED safe in the knowledge that they are the only CICS user task running at that instance.**
- **Field CSACDTA in the CICS CSA is a field that relies on quasi-reentrancy. It points to the TCA of the currently dispatched CICS task. It only has meaning when running under the QR TCB. In CICS TS 3.1 all CICS use of CSACDTA was removed and the field renamed CSAQRTCA.** In the next CICS release beyond CICS TS 3.1 the field will be loaded with a fetch protected address. If used, an abend ASRD will result.

© 2009 IBM Corporation

Terminology

- Open TCBs
  - A new class of CICS TCB available for use by applications
  - Each TCB is for the sole use of the owning CICS task but can be reused by a later task.
    - No subdispatching under Open TCBs, blocking by applications allowed
  - There are several different types or modes of Open TCB.
  - CICS dispatcher domain manages a pool of TCBs for each mode
  - CICS will switch between an Open TCB and the QR TCB as required

© 2009 IBM Corporation

Notes

- OTE introduces a new class of TCB called an open TCB that can be used by applications. An open TCB is characterised by the fact that once it is assigned to a CICS task there is no subdispatching of other CICS tasks under the open TCB. The application can use it to execute under and issue non CICS api requests that may involve the TCB being blocked, e.g. MVS GETMAIN. Blocking is allowed because only this open TCB is halted, and not the whole of CICS. When the CICS tasks ends, the open TCB can be reused by a different CICS task.

- Within the overall class of open TCB there are various modes of TCB. A mode is bidentified by a two character name.

- CICS dispatcher will manage 'pools' of open TCBs, one for each mode.

© 2005 IBM Corporation

© 2009 IBM Corporation

Terminology

- Threadsafe programs
  - –Are capable of being invoked on multiple TCBs concurrently
  - –Cannot rely on quasi-reentrancy to serialise access to resources and storage
  - –A threadsafe program is one that does not modify any area of storage that can be modified by any other program at the same time and it does not depend on any area of shared storage remaining consistent between machine instrucutions.
  - –Must use serialisation techniques such as compare and swap (CS) or enqueue/dequeue to access shared resources with integrity
  - –All programs accessing a shared resource must be made threadsafe e.g. an existing program's reliance on quasi-reentrancy to serialise accessto the CWA is made invalid if just one other program can run concurrently on another TCB and access the same CWA field.

© 2009 IBM Corporation

Notes

- **The term THREADSAFE is used in preference to the term 'fully MVS reentrant' as the latter term is often misunderstood to just mean that a program is linkedited with the RENT option and hence is read-only and does not overwrite itself.**
- **For a program to be threadsafe it has to behave correctly when invoked concurrently under multiple execution units, i.e. TCBs. If it accesses shared storage for example it must ensure that updates to shared storage are serialised. If the program overwrites itself then this is a kind of shared storage and has to be serialised accordingly.**

© 2009 IBM Corporation

CICS Commands and Threadsafe programs

- Threadsafe CICS commands can run on QR or an open TCB
- Non Threadsafe CICS commands must run on QR TCB
- A threadsafe program can issue both threadsafe and non threadsafe CICS commands
  - A Threadsafe command will not cause a TCB switch
  - A non threadsafe command will cause a switch back to QR if not already on QR
    - There is no data integrity issue, just a performance hit of a TCB switch
- However….a program defined to CICS as threadsafe when its application logic is not compromises the data integrity of its resources eg shared storage (no risk to CICS resources)

© 2009 IBM Corporation

Notes

- CICS api and spi commands that are threadsafe can execute correctly on either QR TCB or an open TCB.

- CICS api and spi commands that are not threadsafe must run on QR TCB to maintain data integrity. CICS will ensure these commands always run on QR TCB but switching back to QR TCB if necessary.

- There is no data integrity issue involved with executing non threadsafe cics commands from threadsafe programs. However they do suffer the performance overhead of a switch back to QR TCB if they had been running on an open TCB.

- An application program or Global user exit that is defined to CICS as threadsafe when it is not threadsafe compromises the data integrity of its own resources eg shared storage (no risk to cics maintained resources like temp storage, vsam files etc). Remember when you define a program as threadsafe to CICS you are saying that the **application logic** is threadsafe (eg cobol logic), not anything to do with what CICS commands it issues.

© 2005 IBM Corporation

© 2009 IBM Corporation

Do not neglect Global User Exits
(they will come back to bite you !)

- GLUEs should be made THREADSAFE else excessive TCB switching will occur
- CICS will switch to QR TCB from an open TCB to invoke a non threadsafe GLUE and back again afterwards
- Particularly important exits that need to be threadsafe are XRMIIN and XRMIOUT on the RMI path, and XEIIN/XEIOUT called for all CICS commands.
  - Check with vendors for threadsafe support
  - Use DFH0STAT to find out whats exits in use and if they are defined threadsafe
- Use XPI ENQUEUE and DEQUEUE to serialise access to shared areas like the Global work area (GWA).
- **Warning:** You can no longer rely on field CSACDTA to access the TCA of the current CICS task - only works when running under QR TCB. In CICS TS 3.2 use of CSACDTA produces an ASRD abend.

© 2009 IBM Corporation

Notes

- **If a Global user exit program is not threadsafe and has to be defined as QUASIRENT, then every time it is invoked from an exit point within threadsafe CICS code, if CICS is currently running on an open TCB, it will have to switch to QR to execute the Global User Exit and then switch back to the open tcb before resuming executing the CICS code. Excessive TCB switching for GLUEs will degrade performance and negate the advantages provided by OTE.**

- **CICS has provided new XPI function to allow GLUEs to serialise access to shared storage such as the Global work area (GWA). For EXEC capable GLUEs, code should be added to a HANDLE ABEND routines to DEQUEUE in error situations. For GLUEs that cannot be EXEC capable, CICS will release any enqueues at task termination in error situations.**

- **Field CSACDTA in the CICS CSA cannot be used to access the TCA of the currently dispatched CICS task. It only applies to the task running on the QR TCB. In CICS TS 3.1 all CICS use of CSACDTA was removed and the field renamed CSAQRTCA. In CICS TS 3.2 accessing CSAQRTCA/CSACDTA will produce as ASRD abend.**
  - The XPI can be used to obtain information about the transaction.

© 2009 IBM Corporation

Steps to take: Ensure the program is read-only

- **DFHSIT Parameter RENTPGM=PROTECT**
  - Specifies you want CICS to allocate the read-only DSAs (RDSA and ERDSA) from read-only key-0 protected storage
    - RDSA for RMODE(24) programs and the ERDSA for RMODE(ANY) programs linked RENT
  - Any attempt to overwrite the program will result in message DFHSR0622 and an ABEND0C4
    - Programs running in key-zero or supervisor state can still overlay RDSA and ERDSA

```
DFHSR0622 An attempt to overwrite the RDSA has caused the abend
          which follows

DFHAP0001 An abend (code 0C4/AKEA) has occurred at offset X'00000ACC‘
          in module SQLSPIN.
```


© 2009 IBM Corporation

Notes

- **Running with RENTPGM=PROTECT is an easy way to find out if your programs really are non self modifying. If the program is linkedited with the RENT option, then CICS will load them into read-only storage. An attempt to overwrite the program will produce an 0C4 abend and a DFHSR0622 message.**

© 2009 IBM Corporation

Example: Customer Threadsafe Problem One

- **CICS Region terminated with DFHKE1800**
  - Prior ABEND0C1 in user program SQLSPIN
- **Transaction SQLS in program SQLSPIN suffered the original abend**
- **SQLSPIN is a COBOL program defined as Threadsafe**
- **SQLSPIN Issued DB2 Select command and called Assembler program ASMSETR**

© 2009 IBM Corporation

Notes

- **The slide documents the symptoms of a real problem reported to CICS L2 Support**

© 2009 IBM Corporation

Customer Threadsafe Problem One – COBOL Program SQLSPIN

```
           MOVE 1 TO COUNTER.
           PERFORM WITH TEST BEFORE
               UNTIL COUNTER > 20
                   EXEC SQL
                       SELECT * INTO :VVEMP FROM DSN8710.EMP
                       WHERE EMPNO = '000990'
                   END-EXEC

                   CALL 'ASMSETR' END-CALL
                   ADD 1 TO COUNTER
           END-PERFORM.
           MOVE 'TRANSACTION COMPLETED.'  TO LOGMSG
           EXEC CICS SEND FROM(LOGMSG) LENGTH(22) ERASE END-EXEC.

       END-MAIN.
```

Note: Static and Dynamic Called programs will run on the current TCB
ASMSETR is a Static Called program and will run on the L8 TCB

© 2009 IBM Corporation

Notes

- **A fragment of the customer's COBOL program is shown. Within a loop of DB2 requests, there is a link to a statically called assembler program called ASMSETR which is linkedited with the cobol program. The cobol program is defined as threadsafe, so after an EXEC SQL request, control will return to the application running on an L8 open tcb.**

© 2009 IBM Corporation

Customer Threadsafe problem one – assembler program ASMSETR

```
ASMSETR  CSECT
         USING *,15
RSA       DC    16F'00'           REGISTER SAVE AREA
BEGIN    DS    0H
         SAVE  (14,12)           SAVE R14 Through R12
         ST    13,RSA+4          STANDARD SAVE AREA FOR R13 RECEIVED
         LA    14,RSA            POINT R14 AT SAVE AREA WITHIN PROGRAM
         ST    14,8(,13)         POST FORWARD RSA POINTER
         LR    13,14             POINT R13 TO CURRENT RSA
         LR    3,15              SET BASE REGISTER 3
         USING ASMSETR,3
*
* Business logic was here.
*
         L     13,4(13)
         RETURN (14,12),T,RC=0   RETURN TO SQLSPIN
FILLER   DC    C'XXXXXXXX'
         END   ASMSETR
```


© 2009 IBM Corporation

Notes

- **The prolog and epilog code from the customer's assembler program ASMSETR is shown. In particular the register save area used by the program is defined inline, and not as part of automatic storage such as DFHEISTG. This means that the program is overwriting itelf.**
- **If two tasks running in parallel on their own TCBs enter ASMSETR, they will overwrite each other's registers. On exit one, task will pick up the wrong set of registers. Chaos will result.**

© 2009 IBM Corporation

Customer Threadsafe Problem One – Detection and Prevention

- **Detection**
  - This problem would have been detected if the load module containing the COBOL and Assembler program had been Link-Edited with the RENT attribute and RENTPGM=PROTECT used
    - Since ASMSETR stores within itself you would receive an ABEND0C4 when trying to update the Read Only DSA
- **Prevention**
  - Translate ASMSETR, place RSA in dynamic storage DFHEISTG
  - Or change the call to ASMSETR to use EXEC CICS LINK and ensure ASMSETR defined as Quasirent
    - Not recommended, will cause lots of TCB switching

© 2009 IBM Corporation

Notes

- **If ASMSETR had been linkedited with the RENT option, and CICS was running with the RENTPGM=PROTECT, then and 0C4 would have occurred when ASMSETR at the time it tried to save its registers.**
- **The Register save area needs to be dynamic storage, such as DFHEISTG that is unique to a CICS task.**
- **If the user wanted to serialise access to ASMSETR by defining it as QUASIRENT, then it would need it own program definition and have to be EXEC CICS LINKed to. This is not recommended in this case as it will reintroduce TCB switching.**

© 2009 IBM Corporation

Things to remember about Static and Dynamic Calls

- A statically called program has no CICS program definition of its own
  - Called on whatever TCB the calling program is running on
  - Called programs's logic needs to be threadsafe if calling program is threadsafe
- A Cobol dynamically called program has a program definition
  - But it is only used to allow LE to load the program.
  - Called program is branch and linked to on the callers TCB
  - Called program's logic needs to be threadsafe if calling program is threadsafe
- When a DB2 call is issued from a static or dynamically called routine, the CONCURRENCY attribute of the calling program is used after the DB2 call completes.
  - CICS has no knowledge of the called routine
  - CICS believes current program is the calling program

© 2009 IBM Corporation

Notes

- *If you define a program with CONCURRENCY(THREADSAFE), all routines which are statically or dynamically called from that program (for example, Cobol routines) must also be coded to threadsafe standards.*

- When an EXEC CICS LINK command is used to link from one program to another, the program link stack level is incremented.

- However, a routine which is statically called, or dynamically called, does not involve passing through the CICS command level interface, and so does not cause the program link stack level to be incremented. With Cobol routines, for a static call, a simple branch and link is involved to an address resolved at linkedit time. For a dynamic call, there is a program definition involved, this is required only to allow Language Environment to load the program. After the load, a simple branch and link is executed. When a routine is called by either of these methods, CICS does not regard this as a change of program. The program which called the routine is still considered to be executing, and so the program definition for that program is still considered to be the current one.

- If the program definition for the calling program states CONCURRENCY(THREADSAFE), the called routine must also comply with this specification. Programs with the CONCURRENCY(THREADSAFE) attribute remain on an open TCB when they return from a DB2 call, and this is not appropriate for a program which is not threadsafe. For example, consider the situation where the initial program of a transaction, program A, issues a dynamic call to program B, which is a Cobol routine. Because the CICS command level interface was not involved, CICS is unaware of the call to program B, and considers the current program to be program A.
- Program B issues a DB2 call. On return from the DB2 call, CICS needs to determine whether the program can remain on the open TCB, or whether the program must switch back to the QR TCB to ensure threadsafe processing. To do this, CICS examines the CONCURRENCY attribute of what it considers to be the current program, which is program A. If program A is defined as CONCURRENCY(THREADSAFE), then CICS allows processing to continue on the open TCB. In fact program B is executing, so if processing is to continue safely, program B must be coded to threadsafe standards.

© 2009 IBM Corporation

Steps to take: Look for use of shared storage with CICS IA

- Typical examples of shared storage are …
  - CWA
  - Global user exit global work areas
  - Storage acquired explicitly by the application program with the GETMAIN SHARED option
- You can check whether your application programs use these types of shared storage by looking for occurrences of the following EXEC CICS commands …
  - ADDRESS CWA
  - EXTRACT EXIT GASET
  - GETMAIN SHARED
  - LOAD PROGRAM (eg using dummy assembler program as shared storage)

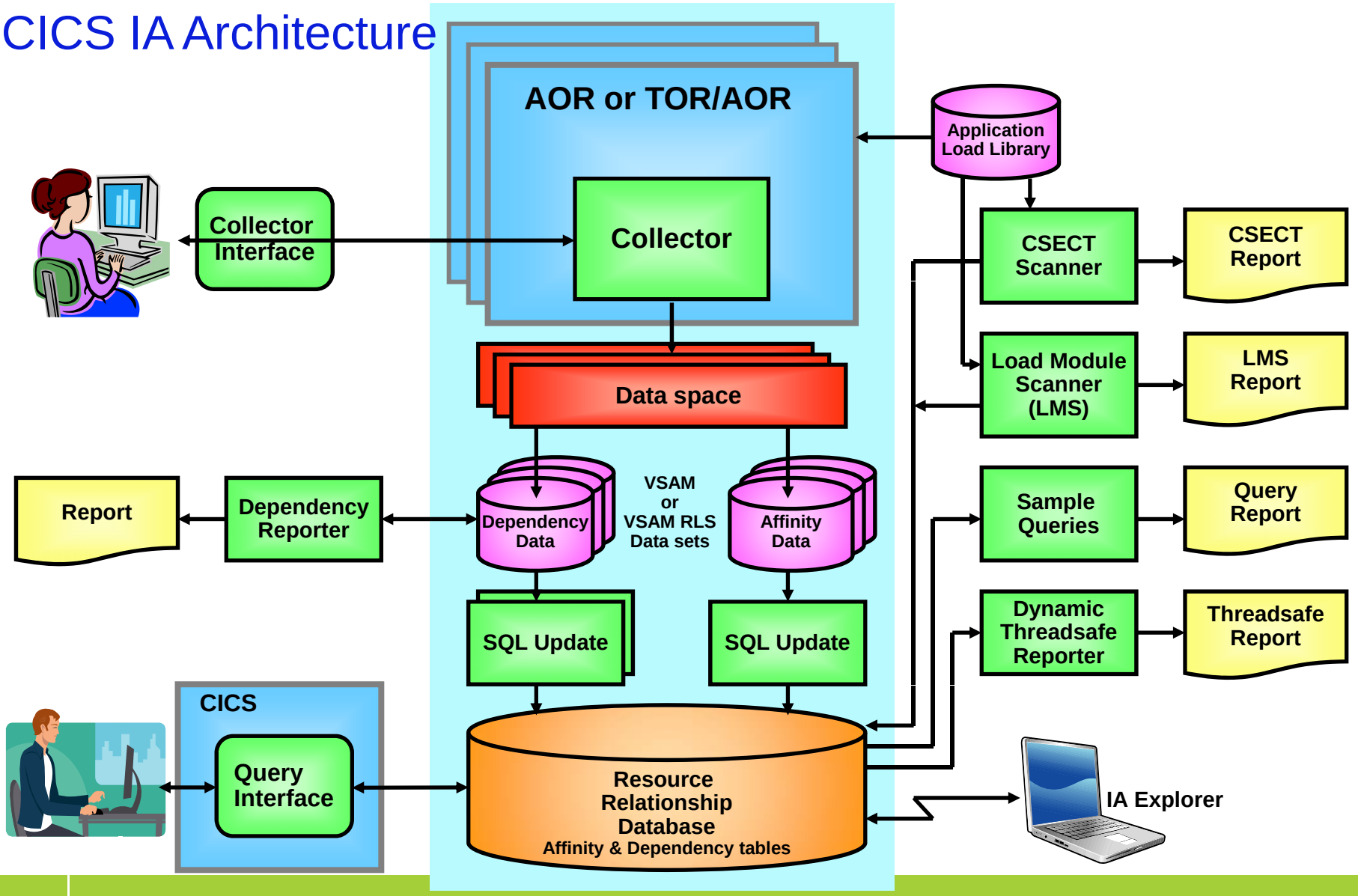
© 2009 IBM Corporation

Notes

- **CICS Interdependency Analyzer (CICS IA) provides the ability not only to scan load libraries for use of commands that give access to shared storage, but can do the same online analysing programs that have actually run. It has much more functionality than the load module scanner in CICS TS.**

© 2009 IBM Corporation

CICS IA Architecture



© 2009 IBM Corporation

Notes

- The Collector is a CICS transaction that runs in your CICS region and intercepts selected (CICS and non-CICS) programming commands. It records details of the resources used by the commands in an MVS data space. This dependency data is subsequently saved to a VSAM file.
- This database contains data extracted from the VSAM dependency file created by the Collector. It is updated periodically to add data from new or infrequently run applications.
- The Query interface is a suite of CICS BMS applications that you can use to access the data held in the database.
- The Dependency Reporter is a batch job that convert the dependency data collected by the Collector into reports in a readable format.
- The Load Module Scanner is a batch utility that scans a load module library to detect those programs in the library that issue commands which may cause transaction resource dependencies. It produces a printed report.
- The CSECT Scanner scans load modules for information that can be used to identify the version of each CSECT. It produces a printed report and creates information that is stored in DB2 tables which can be used, in conjunction with the DB2 dependency tables, to identify different versions of programs
- CICS IA provides a verity of sample SQL queries as a starting point for the user to create there only queries for their specific needs.
- The dynamic Threadsafe reporter creates a report identifying which EXEC CICS commands in the selected applications are threadsafe, non-threadsafe, or indeterminate-threadsafe.
- The IA Explorer is a Eclipse based runtime interface. It helps build queries with which to interrogate the Dependency and Affinity database objects

© 2009 IBM Corporation

Application performance support

- **Results from sample query against the V_CIU_SCAN_TRDSAFE view.**

```
-- q1 - Show all programs with non-threadsafe EXEC CICS commands
**INPUT STATEMENT:
SELECT 'TS31 non-threadsafe calls for ',
       DSNAME, "PROGRAM", OFFSET, COMMAND, ' = ', COUNT(*)
    FROM  V_CIU_SCAN_TRDSAFE
    WHERE (CICS_TS31 = 'N' OR CICS_TS31 IS NULL)
    GROUP BY DSNAME, "PROGRAM", OFFSET, COMMAND;
      +-------------------------------------------------------------------------------------------------------------------+
      |                               |           DSNAME            | PROGRAM  |     OFFSET     | COMMAND  |     |      |
      +-------------------------------------------------------------------------------------------------------------------+
    1_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            685 | INQUIRE  |  =  |    1 |
    2_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            706 | INQUIRE  |  =  |    1 |
    3_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            727 | INQUIRE  |  =  |    1 |
    4_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            748 | DISCARD  |  =  |    1 |
    5_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            769 | SET      |  =  |    1 |
    6_| TS31 non-threadsafe calls for  | CICSIAD.IA2.TEST.LOADLIB      | AFFTESTZ |            790 | INQUIRE  |  =  |    1 |
      +-------------------------------------------------------------------------------------------------------------------+
                                      . . .
```


© 2009 IBM Corporation

Notes

- **This slide shows the result of a query against the V_CIU_SCAN_TRDSAFE table. The report shows what programs contain non-threadsafe EXEC CICS commands.**

© 2009 IBM Corporation

Threadsafe Dynamic Analysis report

```
1CICS INTERDEPENDENCY ANALYZER VERSION 2.2.0                                    2007/10/03:14.01.34    PAGE     1
 Program Dynamic Analysis - THREADSAFE DETAIL LISTING FOR CICS TS

 Report options:
   PROGRAMNAME=*         REGIONNAME=*         CICSLEVEL=      REPORT=DETAIL   LINESPERPAGE=60


 Definitions of Terms:

   'Threadsafe' calls are EXEC CALLS commands that do not cause a TCB swap.

   'Non-Threadsafe' calls are EXEC CALLS commands that cause a TCB swap.

   'Indeterminate Threadsafe' calls are EXEC CALLS commands where it cannot be determined if the call causes a TCB swap.

   'Dynamic calls' are calls to modules at execution time. Programs that are called dynamically take on the same environment
                   as the calling program.

   'Threadsafe Inhibitor calls' are EXEC CICS commands that need to be investigated further because they may prevent you from
                                defining your program as threadsafe. These commands are: ADDRESS CWA, EXTRACT EXIT, GETMAIN SHARED,
                                and LOAD.
```


© 2009 IBM Corporation

Notes

- **The first page of the Dynamic Analysis Threadsafe report shows the options used to create the report and list the definitions for terms used in the report.**

© 2009 IBM Corporation

Threadsafe Dynamic Analysis report - Summary

```
CICS INTERDEPENDENCY ANALYZER VERSION 2.2.0                                    2007/10/19:11.50.59    PAGE     3
Program Dynamic Analysis - THREADSAFE SUMMARY LISTING FOR CICS TS 3.2


APPLID   Program   Linkedit   Execution  Concurrency  APIST  Storage  CICS  LIB Dataset Name
                     Date        Key                        Protect  Rel
-------- -------- ---------- ---------- ---------- ------- -------- ----- --------------------------------

IYDZZ328 EMSCONTA ---------- USER       QUASIRENT   CICSAPI ACTIVE   0650  CICSIAD.TEST.LOADLIB
  Total CICS calls:         13  Threadsafe:            1  Non-Threadsafe:                    7  Indeterminate Threadsafe:         0
                                DB2 calls:             0  MQ calls:                          0  IMS calls:                        0
                                Dynamic Calls:         0  Threadsafe Inhibitor calls:        0

IYDZZ328 EMSTESTS ---------- USER       QUASIRENT   CICSAPI ACTIVE   0650  CICSIAD.TEST.LOADLIB
  Total CICS calls:         64  Threadsafe:           34  Non-Threadsafe:                   26  Indeterminate Threadsafe:         0
                                DB2 calls:             0  MQ calls:                          0  IMS calls:                        0
                                Dynamic Calls:         0  Threadsafe Inhibitor calls:        5
```


© 2009 IBM Corporation

Notes

- **The is a summary listing of the Dynamic Analysis report. It shows the that program EMSTESTS has 34 EXEC CICS commands that will not cause a TCB swap if ran in CICS TS V3.2**
- **It also informs us that there are 5 EXEC CICS calls that inhibit us from going to threadsafe.**
- **The heading of this report indicates the report is run for CICS TS 3.2.**

© 2009 IBM Corporation

Threadsafe Dynamic Analysis report

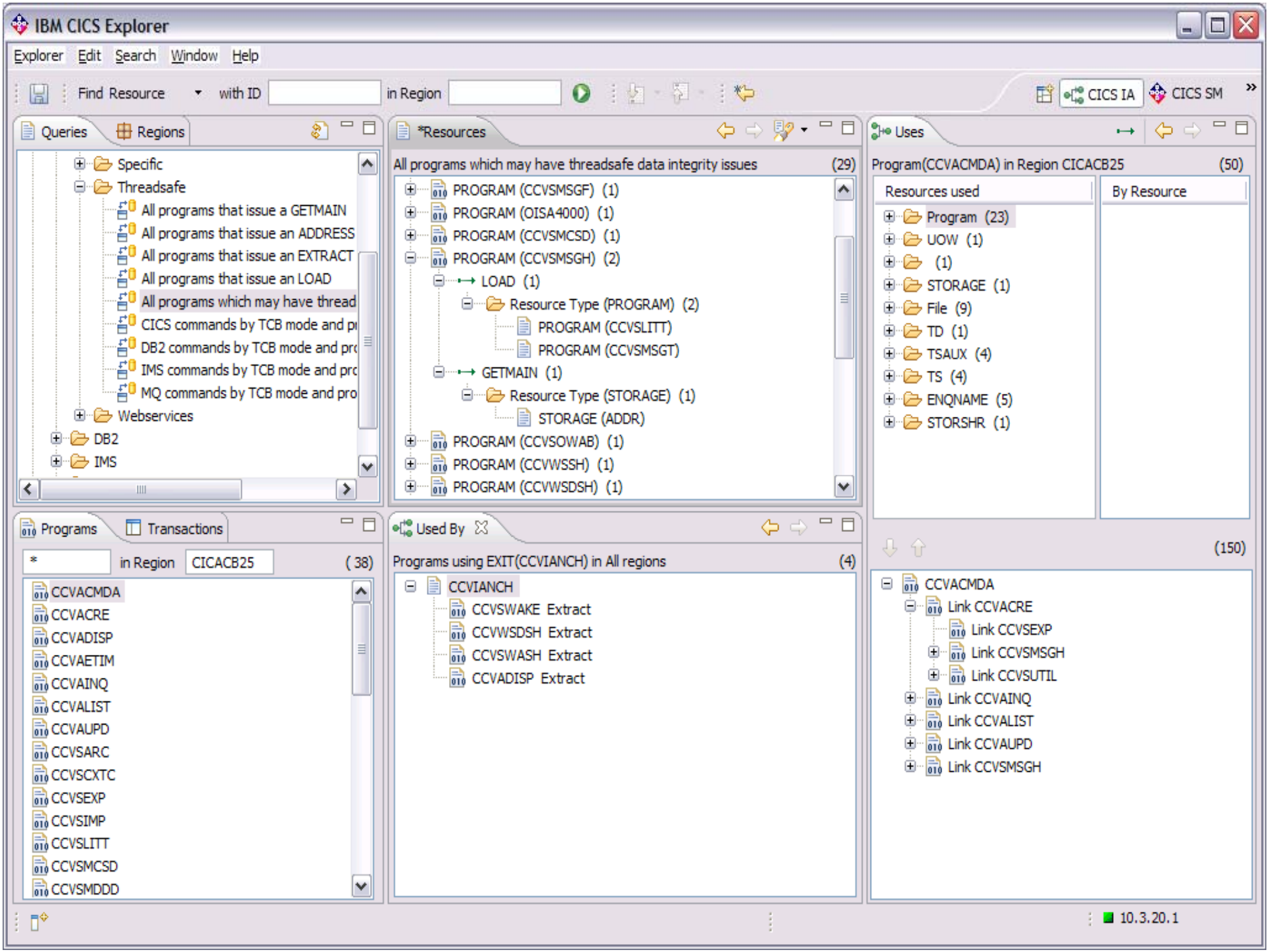
```
CICS INTERDEPENDENCY ANALYZER VERSION 2.2.0                                    2007/10/19:12.22.05    PAGE     3
Program Dynamic Analysis - THREADSAFE DETAIL LISTING FOR CICS TS 3.2

APPLID   Program   Linkedit   Execution  Concurrency  APIST  Storage  CICS  LIB Dataset Name
                     Date        Key                        Protect  Rel
-------- -------- ---------- ---------- ---------- ------- -------- ----- --------------------------------

                 CMD  Function                Type       Resource                        Offset   Program  Use       Threadsafe
                 Type                                                                              Length   Count
                 ---- ----------------------- ---------- ------------------------------- -------- -------- --------- ----------

IYDZZ328 EMSTESTS ---------- USER        QUASIRENT   CICSAPI ACTIVE   0650  CICSIAD.TEST.LOADLIB
                 CICS ADDRESS                         CWA                                  E76     3380         1    Y *
                 .                                    .
                 CICS ADDRESS                         TCTUA                                E76     3380         1    Y
                 CICS ADDRESS                         TCTUA                               1152     3380         1    Y
                 .                                    .
                 .                                    .
                 CICS DEFINE                 COUNTER    TESTDCOUNTER                     1BE0     3380         1    N
                 CICS DELETE                 COUNTER    TESTCOUNTER                      1F2C     3380         1    N
                 CICS DELETE                 COUNTER    TESTDCOUNTER                     1F78     3380         1    N
 Total CICS calls:         64  Threadsafe:           34  Non-Threadsafe:                  26  Indeterminate Threadsafe:
                               DB2 calls:             0  MQ calls:                         0  IMS calls:
                               Dynamic Calls:         0  Threadsafe Inhibitor calls (*):   5
```


© 2009 IBM Corporation

Notes

- **Shows the detail report for program EMSTESTS.**
- **Note: ADDRESS CWA is an inhibitor and is marked with an *.**

© 2009 IBM Corporation


IBM CICS Explorer
Explorer Edit Search Window Help
Find Resource with ID in Region
CICS IA
CICS SM
Queries
Regions
Specific
Threadsafe
All programs that issue a GETMAIN
All programs that issue an ADDRESS
All programs that issue an EXTRACT
All programs that issue an LOAD
All programs which may have thread
CICS commands by TCB mode and pr
DB2 commands by TCB mode and pro
IMS commands by TCB mode and pro
MQ commands by TCB mode and pro
Webservices
DB2
IMS
*Resources
All programs which may have threadsafe data integrity issues (29)
PROGRAM (CCVSMSGF) (1)
PROGRAM (OISA4000) (1)
PROGRAM (CCVSMCSD) (1)
PROGRAM (CCVSMSGH) (2)
LOAD (1)
Resource Type (PROGRAM) (2)
PROGRAM (CCVSLITT)
PROGRAM (CCVSMSGT)
GETMAIN (1)
Resource Type (STORAGE) (1)
STORAGE (ADDR)
PROGRAM (CCVSOWAB) (1)
PROGRAM (CCVWSSH) (1)
PROGRAM (CCVWSDSH) (1)
Uses
Program(CCVACMDA) in Region CICACB25 (50)
Resources used
By Resource
Program (23)
UOW (1)
(1)
STORAGE (1)
File (9)
TD (1)
TSAUX (4)
TS (4)
ENQNAME (5)
STORSHR (1)
Programs
Transactions
* in Region CICACB25 (38)
CCVACMDA
CCVACRE
CCVADISP
CCVAETIM
CCVAINQ
CCVALIST
CCVAUPD
CCVSARC
CCVSCXTC
CCVSEXP
CCVSIMP
CCVSLITT
CCVSMCSD
CCVSMDDD
Used By
Programs using EXIT(CCVIANCH) in All regions (4)
CCVIANCH
CCVSWAKE Extract
CCVWSDSH Extract
CCVSWASH Extract
CCVADISP Extract
(150)
CCVACMDA
Link CCVACRE
Link CCVSEXP
Link CCVSMSGH
Link CCVSUTIL
Link CCVAINQ
Link CCVALIST
Link CCVAUPD
Link CCVSMSGH
10.3.20.1

Notes

- **Example screenshot from the CICS IA plugin to the IBM CICS Explorer, showing built in queries to look for programs that may have access to shared storage.**

© 2009 IBM Corporation

Problem 2 – Non serialized update of the CWA

```
DFHEISTG DSECT
         EXEC SQL INCLUDE SQLCA
EMPNUM    DS   CL6
BONUSLNG  DS   0CL8
BONUSFIL  DS   CL5
BONUSPAK  DS   CL3
CWAREG   EQU  9
SQDWSREG EQU  8
SQDWSTOR DS   (SQLDLEN)C     RESERVE STORAGE TO BE USED FOR SQLDSECT
CWAMAP   DSECT
BONUSTOT DS     XL4
SQLASM1  CSECT
*
         MVC   EMPNUM,=C'000140'
*
* SQL WORKING STORAGE
         LA    SQDWSREG,SQDWSTOR   GET ADDRESS OF SQLDSECT
         USING SQLDSECT,SQDWSREG   AND TELL ASSEMBLER ABOUT IT
*
         EXEC SQL                                                    *
               DECLARE DSN8710.EMP TABLE (                           *
               EMPNO                      CHAR(6),                   *
               SALARY                     DECIMAL,                   *
               BONUS                      DECIMAL,                   *
*
 EXEC SQL SELECT BONUS INTO :BONUS FROM DSN8710.EMP WHERE            *
                 EMPNO= :EMPNUM
*
         XC    BONUSLNG,BONUSLNG
         MVC   BONUSPAK,BONUS
*
         EXEC CICS ADDRESS CWA(CWAREG)
         USING CWAMAP,CWAREG
*
NEWTOTAL EQU   6
*
         CVB   NEWTOTAL,BONUSLNG    Get current bonus in register.
         A     NEWTOTAL,BONUSTOT    Add to old bonus total.
         ST    NEWTOTAL,BONUSTOT    Update CWA with new total.
*
         EXEC CICS RETURN
         END
```


© 2009 IBM Corporation

Notes

- **A second customer problem reported to CICS Level 2 support involved a CICS application used to calculate staff bonuses. The program had recently been defined as threadsafe.**
- **Unfortunately, the program updated data held in the CWA (that is the total amount of bonuses) without any serialisation logic. The bonus total was corrupted.**

© 2009 IBM Corporation

Threadsafe Problem Two – Detection and Prevention

- **Detection**
  - CICS IA would have shown the use of EXEC CICS ADDRESS(CWA)
- **Prevention**
  - EXEC CICS ENQUEUE and DEQUEUE around the update to the CWA
  - Compare and Swap (CS) or Compare Double and Swap (CDS) for Assembler programs
  - Test and Set (TS) instruction for Assembler programs
  - Change RDO Program definition to Quasirent

© 2009 IBM Corporation

Notes

- **Use of CICS IA would have identified that the program was using the CWA and so threadsafe rollout should have involved inspecting the program to see how it manipulated the CWA.**

© 2009 IBM Corporation

CICS Tools that can help – CICS PA

- **Which TCBs did my transaction use?**
  - How many TCB switches (change modes) occurred?
    - What was the Change Mode delay time?
  - How much Dispatch and CPU time did they use?
  - Performance Summary, List and List Extended Reports, …
  - Sample Report Forms …
    - CPU and TCB Usage, TCB Delays, Change Mode Delays, …
- **Why did my transaction take so long?**
  - Wait Analysis Report, Performance List Reports, …
- **Which Transaction(s) used GETMAIN SHARED?**
- **Where did my transaction go?**
  - Cross-System Report, …
  - Performance List, DB2 and WebSphere MQ Reports, …

© 2009 IBM Corporation

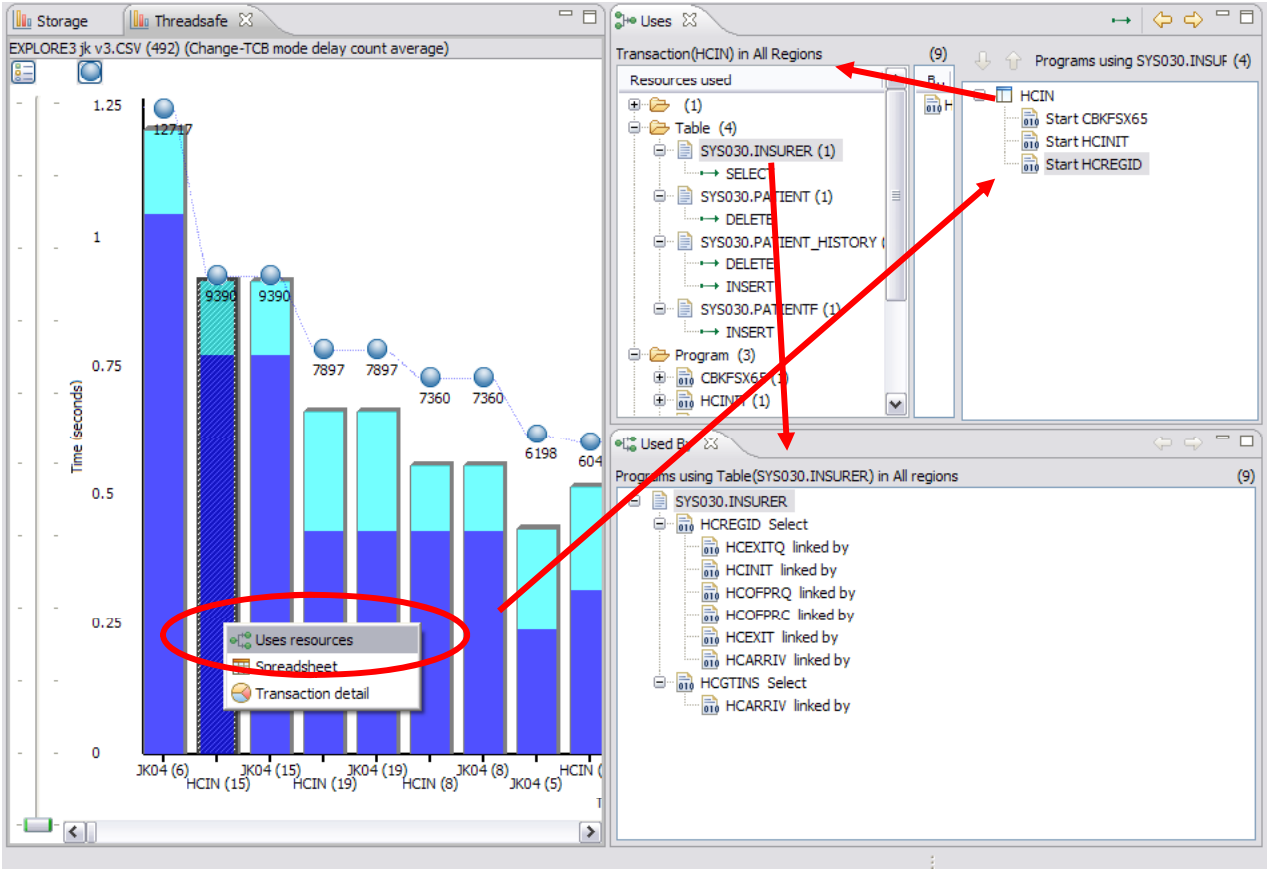
Notes

- **CICS Performance Analyzer (CICS PA) is a powerful tool for unlocking the wealth of information returned by CICS in its SMF 110 performance records. It is easy to see how many TCB switches have occurred for example.**

© 2009 IBM Corporation

Threadsafe (OTE) – Application analysis

- **Use a combination CICS Explorer, CICS IA and CICS PA o see the whole picture**



© 2009 IBM Corporation

Notes

- **Using the CICS IA and CICS PA plugins to the CICS Explorer we can drill down using CICS IA and in context see the relevant performance data**

© 2009 IBM Corporation

Recommended serialisation techniques

- **CICS API: ENQ and DEQ**
  - Advantages:
    - Commands are threadsafe and can be used in all CICS supported languages
    - Application failure will not result in a held lock
    - NOSUSPEND option to get control back if contention
    - No knowledge of assembly language required
  - Disadvantages:
    - Costs more cpu than a non-CICS api technique such as compare and swap
    - Always have to perform ENQ/DEQ even when no contention
    - Must consider implications of MAXLIFETIME option

- **CICS XPI for Global user exits: DFHNQEDX ENQUEUE & DEQUEUE**
  - Same as above.

© 2009 IBM Corporation

Notes

- **The EXEC CICS ENQUEUE and DEQUEUE commands are ideally suited for CICS application programs to serialize access to shared resources. Both commands are threadsafe, so will not incur the performance overhead of switching a task back to the QR TCB.**

- **An enhancement to the exit programming interface (XPI) introduced with CICS Transaction Server 1.3 was the DFHNQEDX macro function call, which provides the same ENQUEUE and DEQUEUE capability provided by the CICS API. The XPI commands are threadsafe, so will not incur the performance overhead of switching a task back to the QR TCB. The XPI ENQUEUE / DEQUEUE is ideal for use within a user exit to serialize access to a global work area (GWA) or any other shared resource. Refer to the *CICS Customization Guide*, SC34-6227 for full details on coding XPI commands.**

© 2009 IBM Corporation

Recommended serialisation techniques

- **Compare and Swap on shared data element**
  - Advantages:
    - Potentially the best performance
  - Disadvantages:
    - Cannot be used for fields greater than 4 bytes (8 bytes for CDS instruction)
    - For fields less than 4 bytes activity on adjacent bytes could cause additional attempts
      - Could move shared data element to new fullword
    - Requires assembly language program or subroutine

© 2009 IBM Corporation

Notes

- **Assembler applications and user exits can use one of the conditional swapping instructions, COMPARE AND SWAP (CS) or COMPARE DOUBLE AND SWAP (CDS) to serialize access to shared resources. Refer to the appropriate Principles of Operation manual for full details on coding these instructions.**

© 2009 IBM Corporation

Recommended serialisation techniques

- **Compare and Swap or Test and Set on separate “lock” byte”**
  - Advantages:
    - “Lock” may be defined for non-contiguous areas
    - If using CS, “locked” status could be task number, terminal id
      - Identifies who owns the lock
  - Disadvantages:
    - Application failure while holding a lock will cause other TCBs to spin until the lock is manually cleared
      - Can avoid the spin by using a lock retry counter but access still denied until lock released
    - Requires assembly language program or subroutine

© 2009 IBM Corporation

Notes

- **An alternative to comparing and swapping individual fields within a shared area, is to make programs access a lock byte first before touching the rest of the share data. A problem with this approach is that an abending transaction may leave the data "locked".**

© 2009 IBM Corporation

Example of COMPARE and SWAP on a shared element

```
* Increment a 4-byte field. ROLD and RNEW are two general purpose regs
INC      DS   0H
         L    ROLD,SHARED            Get  SHARED data element
RETRY    DS   0H
         LR   RNEW,ROLD              Get old value of SHARED
         LA   RNEW,1(,RNEW)          increment value
         CS   ROLD,RNEW,SHARED       Update SHARED element
         BNZ  RETRY                  Update failed, start again
         B    RETURN                 Update worked
```


© 2009 IBM Corporation

Notes

- **The following discussion is predicated on the assumption most accesses to shared resources are for maintaining flags, counters, or chain pointers. In general where this assumption applies, it may be possible to implement a single subroutine (written in assembly language) which protects the integrity of the shared resources, is generally more efficient than ENQ/DEQ, and insulates the application programmer from the details of implementing Compare and Swap instructions for every shared data element.**
- **Except for the actual operation to be performed (increment, decrement, OR, AND, and so on), most Compare and Swap implementations follow exactly the same pattern. The example shows how to increment a 4-byte counter.**
- **Retrying the operation without embedding some form of delay may be disconcerting to some in that it looks as though there is a high potential for a CPU loop. This point is addressed in *ESA/390 Principles of Operation*, SA22-7201, and shown in the following note.**
  Note: **This type of a loop differs from the typical "bitspin" loop. In a bit-spin loop, the program continues to loop until the bit changes. In this example, the program continues to loop only if the value does change during each iteration. If a number of CPUs simultaneously attempt to modify a single location by using the sample instruction sequence, one CPU will fall through on the first try, another will loop once, and so on until all CPUs have succeeded.**

© 2009 IBM Corporation

Example of COMPARE and SWAP on a shared element with retry count

```
* Increment a 4-byte field. ROLD, RNEW and RCOUNT are three general purpose regs
* Abort if retries exceed maxtries
INC      DS   0H
         XR   RCOUNT,RCOUNT       Clear retry count
         L    ROLD,SHARED         Get  SHARED data element
RETRY    DS   0H
         LA   RCOUNT,1(,RCOUNT)   Increment retry counter
         CL   RCOUNT,MAXTRIES     too many attempts?
         BNL  ERROR                yes quit trying
         LR   RNEW,ROLD           Get old value of SHARED
         LA   RNEW,1(,RNEW)       increment value
         CS    ROLD,RNEW,SHARED   Update SHARED element
         BNZ  RETRY               Update failed, start again
         B    RETURN               Update worked
ERROR    DS   0H
         < too many retry attempts >
```


© 2009 IBM Corporation

Notes

- **Implementing a retry counter mitigates the worry of a loop. A retry counter also provides a convenient method for tracking potential resource contention at a very granular level: simply log the retry count somewhere such as in a CICS trace or monitor entry for offline analysis. Adding a retry counter in the code is shown above. The symbol RCOUNT is a register other than ROLD or RNEW.**

© 2009 IBM Corporation

Non-recommended serialisation techniques

- **LINK to a QUASIRENT program**
  - Adds TCB switches, degrades performance
- **CICS transaction class**
  - Maxactive(1) very crude, severe impact on response times
- **MVS enqueue/dequeue**
  - Issuing MVS services disallowed (until openapi in CICS TS 3.1)
  - Threadsafe CICSAPI programs cannot control when a switch to QR may occur

© 2009 IBM Corporation

Notes

LINK to a QUASIRENT program

- **A linked-to program defined as QUASIRENT runs under the QR TCB, and can therefore take advantage of the serialization provided by CICS quasi-reentrancy. Remember even in quasi-reentrant mode, serialization is provided only for as long as the program retains control and does not wait.**
- **A valid serialization technique is therefore to move all shared resource access to a single program, define it as quasi-reentrant. All other application programs can then be defined as threadsafe, on condition they always link to the quasi-reentrant program to access the shared resource.**
- **Although this technique is valid, in that it will protect the integrity of the shared resource, it will not result in the same performance gain as one of the recommended techniques, such as enqueue / dequeue. Whereas the recommended techniques will allow the program to remain on an open TCB (assuming it is there already), this technique will incur the performance overhead of a TCB switch to QR.**

CICS transaction class

- **User defined CICS transaction classes (TRANCLASS) allow the systems programmer to limit the number of concurrent tasks for transactions which belong to the each class. Creating a transaction class with a MAXACTIVE value of 1 is a very crude method of serializing resource access - all transactions belonging to the class will be single threaded. This technique has one advantage in that it can be achieved without changing any application code. However, even in moderately busy system, it is likely to have a severe impact on transaction response times, and runs contrary to the whole objective of implementing threadsafe applications in the first place, that is, improved performance.**

MVS enqueue /dequeue

- **Issuing non CICS API calls from a CICS program is not supported in releases of CICS Transaction Server up to and including Version 2 Release 3, because CICS cannot guarantee such calls will not be issued from QR TCB.**

© 2009 IBM Corporation

Application Design Considerations

- **An ideal threadsafe CICS-DB2 application:**
  - Uses only threadsafe CICS commands
  - All GLUEs on its execution path are threadsafe
  - Switches to L8 once, and stays there for its duration
    - Avoids TCB switching giving cpu decrease
    - Avoids running on QR giving throughput increase

- **Many applications won't fit this ideal**
  - It is still possible to design applications to minimise TCB switches
  - Do not interleave EXEC SQL requests with non threadsafe CICS commands
    - Place non threadsafe CICS commands before first SQL call or after last SQL

© 2009 IBM Corporation

Notes

- An ideal CICS DB2 application program for OTE is a threadsafe application program, containing only threadsafe EXEC CICS commands, and using only threadsafe user exit programs. An application like this moves to an L8 TCB when it makes its first SQL request, and then continues to run on the L8 TCB through any number of DB2 requests and application code, requiring no TCB switching.
- But what if it is not possible to write the ideal CICS DB2 application program? For example, what if some application data is stored in VSAM files? EXEC CICS file control commands are non-threadsafe.
- Even if a number of application programs are not threadsafe, or programs contain non-threadsafe EXEC CICS commands, it is still possible to design application transactions to minimize the number of TCB switches and obtain the performance benefits associated with running threadsafe.
- The execution path between the first and the last SQL call is key to the performance of a CICS DB2 task running under OTE. It follows, that by placing non-threadsafe commands either prior to the first SQL call, or after the final SQL call, the application transaction will avoid incurring the CPU overhead which placing the same code between SQL calls would incur. So, the example of an application with both DB2 and VSAM data, by designing the transactions so the VSAM and DB2 calls are not interspersed, an application of this nature can at least partially exploit OTE.

© 2009 IBM Corporation

Application Design with CICS TS 3.1 : CICSAPI versus OPENAPI

- CICSAPI - services available today under QR TCB
  - CICS command level application programming interface
  - CICS system programming interface
  - CICS Resource Manager Interface (RMI)
  - CICS Exit Programming Interface (XPI) - for Global User exits
  - Systems Application Architecture (SAA) Common Programming Interfaces
    - CPI-C and CPI-RR
  - LE callable services
- OPENAPI - additional APIs possible under Open TCBs
  - Use of MVS services
  - Use of a specified set of POSIX services via MVS Unix System Services

© 2009 IBM Corporation

Notes

- Cicsapi is the term used to describe the set of services available today prior to OTE.

- Openapi is the term used to describe the set of apis available to a program once it is running on an open TCB. An openapi program can issue any of the services available for a cicsapi program but in addition it is able to issue 'foreign api' calls, like MVS services.

© 2005 IBM Corporation

© 2009 IBM Corporation

OTE Changes in CICS TS 3.1 : OPENAPI Programs

- New API Keyword on program definition
  - Attribute of program. Applies to applications, TRUEs, URMs, PLT (ignored for GLUEs)
  - CICSAPI (the default) means the program only uses CICS permitted interfaces
  - OPENAPI means the program requires an Open TCB to use other APIs (OPENAPI requires CONCURRENCY(THREADSAFE))
- New L9 TCB for running userkey OPENAPI programs
  - For OPENAPI programs TCB key must match PSW key for non-CICS apis to work (whereas CICS APIs run in either key irrespective of TCB key)
  - SIT parm MAXOPENTCBS now covers L8 and L9 TCBs
- New CICS uses of L8 TCBs
  - When accessing DOCTEMPLATEs or static HTTP responses held on HFS
  - WebServices and XML support implemented using cicskey OPENAPI programs

© 2009 IBM Corporation

Notes

- A new API keyword on the program definition tells CICS whether the program is to use CICS apis or whether it is potentially going to use other apis as well as cics apis. The default is CICSAPI. OPENAPI is the trigger to tell CICS that this application **must** run on an open TCB (as opposed to THREADSAFE which says the application is able to run on an open TCB if CICS code or TRUE code is executing on an open tcb at the time control is to be passed back to application code).

- OPENAPI programs must be threadsafe and defined to CICS as such.

- Because OPENAPI programs can potentially use non CICS APIs, the key of the TCB becomes important, and the key of the TCB must match the execution key. This is a contrast with CICSAPI threadsafe programs that can execute in cics key or user key irrespective of the tcb key. This is because CICS services are implemented independent of the key of the tcb they are running on, whereas MVS services for example use the tcb key.

- The pool of open tcbs, whose size is specified via SIT parm MAXOPENTCBS, now contains both L8 and L9 TCBs. Its size needs to be adjusted to cater for L9 TCBs and new CICS uses of L8 TCBs.

- CICS internally now uses L8 TCBs when accessing HFS to retrieve doctemplates stored on HFS, or to retrieve static HTTP responses held on HFS specified via the new URIMAP resources definition. Additionally support for WebServices and XML employ cicskey openapi programs that likewise use L8 TCBs.

© 2005 IBM Corporation

© 2009 IBM Corporation

Notes

- A new API keyword on the program definition tells CICS whether the program is to use CICS apis or whether it is potentially going to use other apis as well as cics apis. The default is CICSAPI. OPENAPI is the trigger to tell CICS that this application **must** run on an open TCB (as opposed to THREADSAFE which says the application is able to run on an open TCB if CICS code or TRUE code is executing on an open tcb at the time control is to be passed back to application code).

- OPENAPI programs must be threadsafe and defined to CICS as such.

- Because OPENAPI programs can potentially use non CICS APIs, the key of the TCB becomes important, and the key of the TCB must match the execution key. This is a contrast with CICSAPI threadsafe programs that can execute in cics key or user key irrespective of the tcb key. This is because CICS services are implemented independent of the key of the tcb they are running on, whereas MVS services for example use the tcb key.

- The pool of open tcbs, whose size is specified via SIT parm MAXOPENTCBS, now contains both L8 and L9 TCBs. Its size needs to be adjusted to cater for L9 TCBs and new CICS uses of L8 TCBs.

- CICS internally now uses L8 TCBs when accessing HFS to retrieve doctemplates stored on HFS, or to retrieve static HTTP responses held on HFS specified via the new URIMAP resources definition. Additionally support for WebServices and XML employ cicskey openapi programs that likewise use L8 TCBs.

© 2005 IBM Corporation

© 2009 IBM Corporation

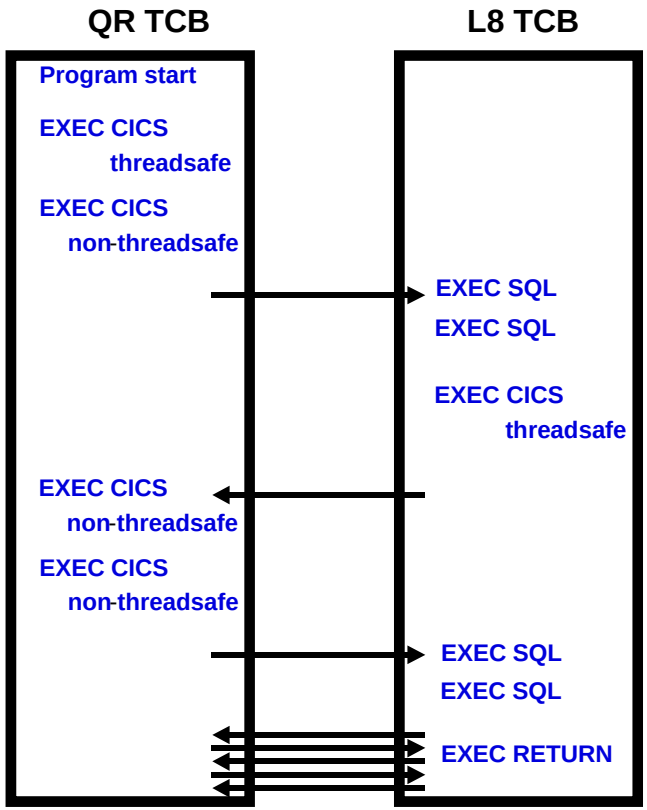
Notes

- OPENAPI programs must be coded to threadsafe standards and defined to CICS as such.
- The primary purpose for support of OPENAPI programs is to allow Application workloads to be moved off QR TCB. This allows better use of machine resources and better throughput can be achieved.
- Openapi programs can use non CICS api requests because they are not running on the QR TCB. However use of non CICS APIs within CICS is entirely at the risk of the user. No testing of non CICS APIs within CICS has been undertaken and is not supported by IBM service.

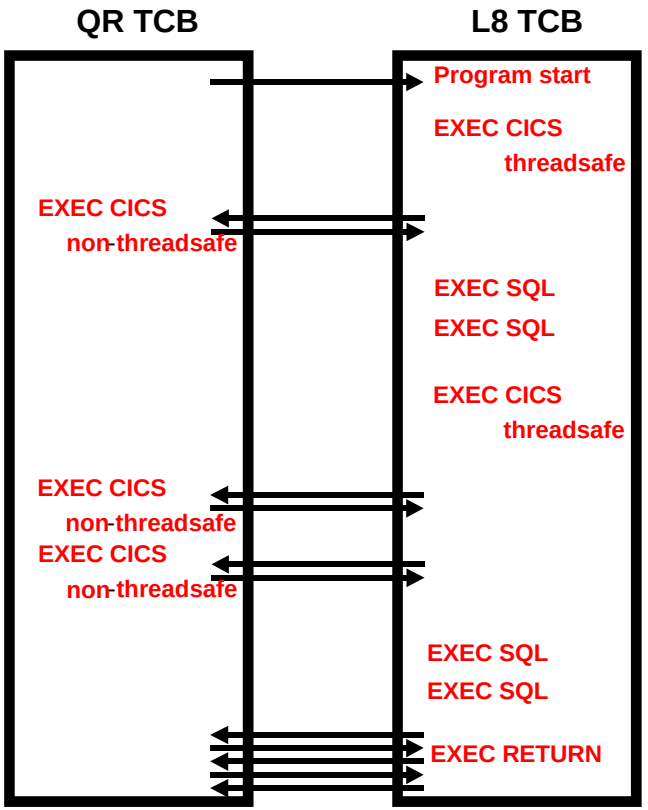
© 2005 IBM Corporation

© 2009 IBM Corporation

CICS TS 3.1 - TCB switching



The program for transaction BLUE is defined THREADSAFE , API= CICSAPI



The program for transaction RED is defined THREADSAFE, API= OPENAPI ,EXECKEY=CICS

© 2009 IBM Corporation

Notes

In CICS Transaction Server R3.1, a program may be defined with the OPENAPI keyword, to indicate it should execute on an OPEN TCB (L8 or an L9, based on the EXECKEY specification). However, it should be noted *all non-threadsafe commands will continue to be processed on the QR TCB.*

On the prior page notice the program used for transaction BLUE is defined as THREADSAFE, with API=CICSAPI. The execution of the task and TCB switching is the same as CICS Transaction Server R2.2 and R2.3. The program executes on the QR TCB until it issues an DB2 request, at which point it will switch to an L8 TCB. It will remain on the L8 until a non-threadsafe command is issued, causing it to switch back to the QR where it will remain until an other SQL request is issued.

The application used for transaction RED is defined THREADSAFE, with API=OPENAPI and EXECKEY=CICS. The program is given control on an L8 TCB. Each time it issues a non-threadsafe command it will switch to the QR TCB to process the command and then return control to the application on the L8 TCB. SQL commands are executed on the L8 TCB along with any threadsafe commands. Notice, if there are many non-threadsafe commands, the overhead of switching can be greater than running with API=CICSAPI.

Another important point to consider. If the program for transaction RED had been defined to run in EXECKEY USER, it will be given control on an L9 TCB, rather than an L8. The processing will be the same as noted above, EXCEPT when an SQL call is issued, the task is switched to an L8 TCB. Upon completion of the SQL request, control will be returned to the application on the L9 TCB.

© 2005 IBM Corporation

© 2009 IBM Corporation

CICSAPI THREADSAFE versus OPENAPI THREADSAFE

- **NB: OPENAPI TRUEs must run CICS key on an L8 TCB**
  - A user key threadsafe OPENAPI program calling DB2 will switch from L9 to L8 then back to L9 for each DB2 request. Same applies to WMQ and sockets
  - A user key threadsafe CICSAPI program running on an L8 TCB will remain on the L8 TCB to call DB2 or WMQ or sockets.

- **Candidates for CICSAPI with THREADSAFE**
  - SQL programs with some non-threadsafe API
  - SQL programs with USER key
  - Mixed FC and DB2 and/or WMQ and/or socket applications
    - FC relies on the DB2 or WMQ or sockets call to push the task onto the L8 TCB

- **Candidates for OPENAPI with THREADSAFE**
  - programs with threadsafe APIs only including FC RLS and FC Local LSR
  - SQL programs with CICS key
  - CPU intensive programs

© 2009 IBM Corporation

Notes

- It should be noted that use of OPENAPI programs can increase TCB switching. For example Task Related User Exits (TRUEs) have to run in CICS key on L8 TCBs. Hence if an openapi user key application calls DB2, then it will switch from the L9 TCB to the L8 TCB to call DB2, and then back to L9 to return to the application. **User key DB2 applications are best left defined as CICSAPI Threadsafe applications.**

- CICS key openapi programs will receive control on an L8 TCB and no switching will occur when calling an OPENAPI TRUE.

- If a non threadsafe CICS command is issued from an OPENAPI program, then CICS switches to QR TCB to execute the CICS request, and then switches back to the open tcb when returning to the application program. This is because when you define a program as openapi you are saying that it **must** run on an open tcb. Whereas a cicsapi threadsafe application is one that is is happy to run on whatever tcb cics deems fit. In this case a non threadsafe cics command will cause a switch to QR TCB, but then control will remain on the QR TCB when we return to the application, until something happens that forces a switch back to the open tcb, eg a DB2 call. Hence use of non threadsafe cics commands will cause more tcb switching for a OPENAPI threadsafe program than a CICSAPI threadsafe program.

© 2005 IBM Corporation

© 2009 IBM Corporation

Further information

- CICS TS 2.3/3.1/3.2/4.1 Information Centers
  - Refreshed regularly, can be downloaded from:
  - http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi
- Redbook: Threadsafe considerations for CICS SG24-6351-00
  - http://www.ibm.com/redbooks
  - Third edition (November 2007) incorporates CICS TS 3.2 enhancements
- CICS Tools to help threadsafe migration
  - http://www.ibm.com/cics/tools

© 2009 IBM Corporation