CICS and Threadsafe

Exploiting the Open Transaction Environment

Russ Evans
russ@evansgroupconsulting.com
Objectives

- History of Multithreading
- The Open Transaction Environment
- Making programs Threadsafe
- Exploiting the OTE
- OTE Performance Considerations
- Diagnosing Threadsafe Problems
- Recommendations
History of Multithreading

• CICS as a Single TCB
  – Most efficient on a uni-processor
  – “Quasi-Reentrancy”
  – Issues:
    • Runaway tasks
    • OS Waits = Region Wait
    • Many restricted OS and COBOL Commands
    • Limited by speed of one processor
History of Multithreading

• CICS Exploiting Multiple Processors
  – Multiple TCBs
  – Primary TCB is “QR”, Quasi-Reentrant
  – Additional TCBs for:
    • VSAM
    • DB2
    • Program Loader
    • etc.
History of Multithreading

• CICS and DB2
  – Separate TCB (‘thread’) for each DB2 Request
  – Task is switched to DB2 TCB for DB2 work, DB2 system code runs on DB2 TCB
  – Significant workload shifted to DB2 TCBs, but measurable overhead from TCB switching
Open Transaction Environment

- Transaction runs under own TCB
- Introduced in TS 1.3 for Java
- DB2 Support added for TS 2.2
- Supports full OS function
- Allows true Multitasking in CICS
- Pseudo-reentrancy no longer allowed
OTE and DB2

Without Threadsafe

QR TCB

Task Starts

EXEC CICS

EXEC SQL

Application Code

EXEC SQL

Open TCB

DB2 Code executes

DB2 Code completes

DB2 Code executes

DB2 Code completes
OTE and DB2

With Threadsafe

**QR TCB**

Task Starts

**EXEC CICS**

**EXEC SQL**

DB2 Code executes

Application Code

DB2 Code executes

Task Termination 

Task completes
So, What’s the Problem

CICSRGN1

TASK1

PROG001

MOVE CWA-COUNTER TO OUTPUT-FIELD
ADD +1 TO CWA-COUNTER
EXEC CICS WRITE OUTPUT-RECORD

CWA

0001

0001 + 1 = 0002

stuff0001morestuff
So, What’s the Problem

CICSREGION1

**TASK1**

**PROG001**

MOVE CWA-COUNTERTO OUTPUT-FIELD
ADD +1 TO CWA-COUNTER
EXEC CICS WRITE OUTPUT-RECORD

stuff0001morestuff

0001 + 1 = 0002

**CWA**

0001

**TASK2**

**PROG001**

MOVE CWA-COUNTERTO OUTPUT-FIELD
ADD +1 TO CWA-COUNTER
EXEC CICS WRITE OUTPUT-RECORD

stuff0001morestuff

0002 + 1 = 0003

Copyright (c) 2013, The Evans Group, Inc.
Definitions

Define “threadsafe”

1. “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 does not depend on any area of shared storage remaining consistent between machine instructions.”
Controlling Threadsafe

• At the program level:
  Parameter on Program Definition
  • CONCURRENCY=QUASIRENT (Not Threadsafe)
  • CONCURRENCY=THREADSAFE
  • CONCURRENCY=REQUIRED

• At the region level, new SIT parm:
  FORCEQR=YES/NO
  • FORCEQR=YES  All programs run non-Threadsafe
  • FORCEQR=NO  Programs follow CONCURRENCY parm on program definition
Identifying Threadsafe Programs

• No automated method of identification
• IBM Tool can help
• Rules of thumb:
  – COBOL and PL/1 must be LE
  – All programs must be re-entrant
  – Aps with no affinities are more likely to be threadsafe
Identifying Threadsafe Programs

Ensure programs are re-entrant:

• **COBOL:**
  – Compile with RENT
  – Link with RENT

• **Assembler:**
  – Code review, possible coding changes required
  – Assemble/Link with Rent

• **CICS:**
  – RENTPGM=PROTECT
  – Adjust RDSA/ERDSA sizes
  – Non-reentrant activity will generate DFHSH0622 followed by S0C4/ASRA
  – Possible conflicts with debuggers
Identifying Threadsafe Programs

No automated method of identification

CONCURRENCY
parm is a
promise
by you, not an order to CICS
Definitions

Define “threadsaf e”

1. “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 does not depend on any area of shared storage remaining consistent between machine instructions.”

2. “A program defined as CONCURRENCY=THREADSAFE is one that will be allowed to run on an open TCB.”
Identifying Threadsafe Programs

Continued...

There is a tool available to help start.....

• Utility DFHEISUP will scan for CICS commands commonly used in non-threadssafe applications
• Use command table DFHEIDTH
Making Programs Threadsafe

After identifying non-Threadsafe code you have two choices:

1) Alter the code to serialize the shared storage access
   A) Use CICS to automatically ensure serialization
   B) Manually ensure serialization

2) Do nothing
If shared storage use is limited to few programs:

- Leave non-threadsafe programs QUASIRENT
- CICS will switch to QR on LINK or XCTL (But…**not for CALL**!)
- Access to shared storage is automatically serialized
Making Programs Threadsafe

Leave Program CONCURRENCY(QUAISRENT)

CICSRGN1

**TASK1**

**PROG001**
- MOVE CWA-COUNTER TO OUTPUT-FIELD
- ADD +1 TO CWA-COUNTER
- EXEC CICS WRITE OUTPUT-RECORD

**CWA**
- 0001

**stuff0001morestuff**

**TASK2**

**PROG001**
- Wait For QR

0001 + 1 = 0002

stuffed0001morestuffed
Advantages:
• No coding changes, so quick implementation

Disadvantages:
• Additional TCB switching overhead
• Maintenance issues
• All programs that access these areas **must** also remain QUASIRENT
Making Programs Threadsafe

To serialize access to shared storage:

- “Wrap” access in CICS ENQ/DEQ
- For Assembler, use CS/CDS
- Move data to a threadsafe but serialized facility:
  - CICS Maintained Data Table
  - DB2 table
  - Coupling Facility
- Force single-threading with TCLASS??
Making Programs Threadsafe

CICSRGN1

TASK1
PROG001
EXEC CICS ENQ()
MOVE CWA-COUNTER TO OUTPUT-FIELD
ADD +1 TO CWA-COUNTER
EXEC CICS DEQ()
EXEC CICS WRITE OUTPUT-RECORD
stuff0001 morestuff

0001 + 1 = 0002

CWA

0001

0002 + 1 = 0003

TASK2
PROG001
EXEC CICS ENQ()
...wait on enq...
MOVE CWA-COUNTER TO OUTPUT-FIELD
ADD +1 TO CWA-COUNTER
EXEC CICS DEQ (*)
EXEC CICS WRITE OUTPUT-RECORD
stuff0002 morestuff

stuff0001

stuff0002
Making Programs Threaadsafe

ENQ Issues:

• CPU Cost
• Potential bottleneck
  – Limit ENQ duration by issuing DEQ as soon as possible
  – Ensure no possibility of deadly embrace
Regardless of which method, remember:

All programs that access the same shared storage area in the same CICS region **must** be converted before **any** of these programs are marked as Threadsafe!
Accessing The OTE

Three methods of executing on OTE TCB

• Create a dummy OPENAPI TRUE
• Define program as API(OPENAPI)
• Define program as CONCURRENCY(REQUIRED)
Accessing The OTE

Using a dummy TRUE

For CICS 2.2 and above, write a “dummy” TRUE

- Include OPENAPI on the ENABLE command
- The TRUE program **must** be defined as Threadsafe
- See the CICS Customization Guide section on Task Related User Exits
Accessing The OTE

Functions like DB2 call:

• When task calls OPENAPI true, spun to L8 TCB
• If user program THREADSAFE, task remains on L8 until forced off
• L8 TCB owned until task termination
• No supported method to tell if task is on L8 or QR
• **Review restrictions defined in Customization Guide!**
Accessing The OTE

DMYRMCAL TITLE ' - Sample Dummy stub for TRUE for OPENAPI Processing.*
**-----------------------------------------------------------------------------**
** Name    : DMYRMCAL
** Purpose : Provide a means to programmatically force a task to *
** be spun to an L8 TCB. *
** This is the callable stub that invokes the dummy *
** TRUE. This stub must be linked into any program *
** wishing to use the TCB spin TRUE. It is called via *
** standard call syntax: *
** CALL DMYRMCAL *
** As no actual work is performed by the TRUE, no parms*
** are used on the call statement. *
** *
**-----------------------------------------------------------------------------**
**
** ** Module entry point.
DMYRMCAL CSECT ,          Define the module environment
DMYRMCAL AMODE 31
DMYRMCAL RMODE 31
DFHRMCAL TO=DMYTRUE       Call the TRUE
LTORG ,
END   DMYRMCAL
Accessing The OTE

DMYTRUE TITLE ' - Sample Dummy TRUE for OPENAPI Processing.'
**-----------------------------------------------**
** Name   : DMYTRUE    **
** Purpose : Provide a means to programmatically force a task to **
**           be spun to an L8 TCB.               **
** Returns : Rc in R15 == 0               **
**          **
**-----------------------------------------------**

    DFHUEXIT TYPE=RM          Parmlist is passed in R1
**
**
**----------------------------------------------- Module entry point.   **
DMYTRUE CSECT ,           Define the module environment
DMYTRUE AMODE 31
DMYTRUE RMODE 31
SR    15,15
BR    14
LTORG ,
END    DMYTRUE

Copyright (c) 2013, The Evans Group, Inc.
Accessing The OTE

QR TCB

Task Starts

Non-threadsafe code
E.C. non-threadsafe

CALL ‘DMYRMCAL’

Open TCB

DMYTRUE executes
Threadsafe user code
E.C. threadsafe

E.C. non-threadsafe
Task Termination

E.C non-threadsafe
Accessing The OTE

Returning The Task to QR TCB

- Clone DMYTRUE/DMYRMCAL
- Define DMxTRUE as CONCURRENCY=QUASIRENT
- Enable the new exit as QUASIRENT
Accessing The OTE

QR TCB

Task Starts

Non-threadsafe code
E.C. non-threadsafe

CALL ‘DMYRMCAL’

Open TCB

DMYTRUE executes

Threadsafe user code
E.C. threadsafe

Non-threadsafe code

Task Termination

CALL ‘DMxRMCAL’
Accessing The OTE
OPENAPI

For CICS 3.1 and higher, modify the PROGRAM definition on the application program to API=OPENAPI

- The program **must** be Threadsafe
- **All** application code runs in the OTE environment
- **All** application code runs on the same TCB instance on which the program was initialized.
Accessing The OTE

Forces program to run on L8/9 TCB:

- Program is initialized on L8 TCB if CICS key
- Program is initialized on L9 TCB if USER key
- If program issues non-threadsafe command, task is spun to QR
- Once command has completed, task is spun to L8/9
- Use INQUIRE_CURRENT_PROGRAM and INQUIRE_PROGRAM to identify
Accessing The OTE

QR TCB

Open TCB

Task Starts

E.C. threadsafe

E.C. threadsafe

Command Starts

E.C. non-threadsafe

Command Completes

Task Termination
Accessing The OTE

There are performance issues for USER key OPENAPI programs that also access OPENAPI TRUEs (includes DB2)

• USER key Program is initialized on L9 TCB
• OPENAPI TRUE is initialized on L8 TCB
• When L9 program issues DFHHRMCCAL to OPENAPI TRUE:
  – Task is spun to L8 TCB for duration of TRUE
  – Task is returned to L9 following completion of TRUE
• L8 TCB instance held until task termination
Accessing The OTE

There are performance issues for USER key OPENAPI programs that also access OPENAPI TRUEs (includes DB2)

- Review MAXOPENTCB for possible increase
- Review TCBLIMIT for possible increase
- Open TCB “stealing” performance issues
- Potential TCB deadly embrace
For CICS 4.2, modify the PROGRAM definition on the application program to API(CICSAPI) and CONCURRENCY(REQUIRED)

- **The program** **must** be Threadsafe
- **All** application code runs in the OTE environment
- **All** application code runs on the same TCB instance on which the program was initialized.
- **All** application code runs on an L8 TCB
Accessing The OTE

Forces program to run on L8 TCB:

- Program is initialized on L8 TCB
- If program issues non-threadsafe command, task is spun to QR
- Once command has completed, task is spun to L8
- Use INQUIRE_CURRENT_PROGRAM and INQUIRE_PROGRAM to identify
Accessing The OTE

QR TCB

Open TCB

Task Starts

E.C. threadsafe

E.C. threadsafe

Command Starts

E.C. non-threadsafe

CommandCompletes

Task Termination
Accessing The OTE

There are no additional performance issues for USER key CONCURRENCY(REQUIRED) programs that also access OPENAPI TRUEs (includes DB2)

• USER key Program is initialized on L8 TCB
• OPENAPI TRUE is initialized on L8 TCB
• Only one L8 TCB is acquired by the task
  – L8 is shared by user program and all OPENAPI TRUEs
• L8 TCB instance held until task termination
So........

Which Way Is Best?
Accessing The OTE

Via Dummy TRUE

Advantages:
• Control application environment programmatically
• CPU savings if large number of non-threadsafe commands
• CPU savings when accessing OTE in USER key
• Non-threadsafe application code may continue to run on QR TCB
• Available in CICS 2.2 and above.
Accessing The OTE

Via Dummy TRUE

Disadvantages:

• Requires changes to application code
• Requires process to enable TRUE
• If any non-threadsafe commands, must call TRUE prior to any OTE activity
• Cannot determine environment programmatically
• Probability future maintenance results in inadvertent use of restricted code on QR, data corruption, region failure
Accessing The OTE

Via OPENAPI Parm

Advantages:
- No coding changes required
- All application code guaranteed to run in OTE
- No requirement to enable TRUE
- Can determine environment programmatically
- All user code on same TCB – no issues with “paired” z/OS macros
Accessing The OTE

Via OPENAPI Parm

Disadvantages:

• CPU overhead when accessing OPENAPI TRUE in USER key (DB2, etc.)
• CPU overhead when issuing non-threadsafe EXEC CICS commands
• All application logic must be threadsafe
• Can increase the number of open TCBs required.
• Overhead if TCB stolen to switch key
Accessing The OTE

Via CONCURRENCY(REQUIRED) Parm

Advantages:

• No coding changes required
• All application code guaranteed to run in OTE
• No requirement to enable TRUE
• Can determine environment programmatically
• All user code on same TCB – no issues with “paired” z/OS macros
Accessing The OTE

Via CONCURRENCY(REQUIRED) Parm

Disadvantages:

• CPU overhead when issuing non-threadsafe EXEC CICS commands
• All application logic **must** be threadsafe
Accessing The OTE

Via CONCURRENCY(REQUIRED) with API(OPENAPI)

Disadvantages:
• Can increase the number of open TCBs required.
• Overhead if TCB stolen to switch key
Accessing The OTE

Via CONCURRENCY(REQUIRED)
with
API(CICSAPI)

Disadvantages:
• Limited to using standard CICS services
• Potential problems if unsupported z/OS services used
Accessing The OTE

One restriction in OPENAPI programs:

- Do not attempt to initialize batch LE environment under CICS OPENAPI.
Implications of New TCB Types

• Multiple TCB types
• Application code running in OTE
  – Application programs fighting for CPU
  – Poor coding only affects program user, not region
  – Resource hogs build up
• CICS system code running in multiple TCBs
• IBM converting sub-products to use OTE
  – MQ
  – Sockets
  – XML parser
Why Bother?

Run tasks on an open TCB to:

- Reduce QR CPU constraint by moving tasks to other processors
- Use z/OS functionality forbidden on QR TCB
  - Activity generating z/OS waits
    - I/O
    - ENQ/DEQ
- Segregate troublesome transactions
Reducing QR CPU Constraint

QR TCB is limited to the speed of one processor

When QR hits CPU limit, region stalls

- Classic fixes
  - Clone Region to offload CPU
- Modern fix = Exploit OTE to offload CPU
Reducing QR CPU Blocking

QR TCB is single threaded

- Current task “owns” QR until next EXEC CICS (*)
- Heavy CPU routines don’t release QR
- Region appears to lock up
- While task runs, CICS workload backs up
  - VSAM, DB2 I/O Completes
  - New tasks ready for dispatch
- ....
Reducing QR CPU Blocking

OTE is Multi-Threaded

• OTE task “owns” his TCB until next EXEC CICS (*)
• QR is available for other workload
• No region hold-up
• No extended response times
  • Other workload unaffected
• Response time improves
Reducing QR CPU Constraint

Warning: Consider LPAR CPU Implications when converting a QR constrained region to exploit open TCBs:

- Reduce QR constraint by moving tasks to other processors
- In MP environment, total CICS CPU will increase until:
  
  1. CICS CPU requirements satisfied  
  2. Box CPU capacity met

- Can negatively impact z/OS workload CICS depends on
Multiple TCB Structure

Classic CICS

z/OS

CPU1

CICS/QR
Task1

CPU2

CICS/L8
DB2

CPU3

z/OS
Sockets

CPU4

MQ

Copyright (c) 2013, The Evans Group, Inc.
Multiple TCB Structure

Modern CICS With Threadsafe Applications

z/OS

CPU1
  CICS/L8
  CICS/QR
  Task1

CPU2
  CICS/L8
  Task2

CPU3
  CICS/L8
  CICS/X8
  Task3

CPU4
  CICS/L8
  CICS/J8
  XML

Copyright (c) 2013, The Evans Group, Inc.
Using Forbidden Functionality

Use almost any z/OS function:
- Communicate with operator via WTOR
- Make use of flexibility of STORAGE OBTAIN/RELEASE
- Issue I/O without CICS file control
- Use z/OS ENQ/DEQ to synchronize with batch jobs
- ........
Using Forbidden Functionality

Transaction initiated communication with operator via WTOR:

- OTE TCB waits, not entire region
- Synchronous waits on external events/requests
- CICS command input from master console
- Enable use of standard auto operation facility

Disadvantages:
- Task shows as “running”
- No way to track WTOR back to task
Using Forbidden Functionality

Use of z/OS STORAGE OBTAIN/RELEASE

- Powerful options not available from EXEC CICS GETMAIN
- Storage acquired outside of CICS subpools
- More efficient than CICS GETMAIN

Disadvantages:
- Storage invisible to CICS monitor
- No automatic cleanup at task termination
- Storage not displayed in dump, trace, etc.
- Problems with OS GETMAIN and USER key OPENAPI tasks
Using Forbidden Functionality

Error on STORAGE OBTAIN causes ASRB, not region failure:
DFHAP0001 CICSD225 An abend (code 878/AKEB) has occurred at offset X'FFFFFFFF' in module TEST.

```
00057 L9002 AP 00E1 EIP   EXIT  LOAD
00057 L9002 AP 1942 APLI  *EXC*  Abend
00057 L9002 AP 0791 SRP   *EXC*  MVS_ABEND
00057 L9002 DS 0010 DSBR  ENTRY  INQUIRE_TASK
00057 L9002 DS 0011 DSBR  EXIT   INQUIRE_TASK/OK
00057 QR    PG 0500 PGIS  ENTRY  INQUIRE_CURRENT_PROGRAM
00057 QR    PG 0501 PGIS  EXIT   INQUIRE_CURRENT_PROGRAM
00057 QR    AP 0782 SRP   *EXC*  ABEND_ASRB
```

TCB is marked as unusable:

```
DSTCB QR    KE 0502 KEDS  ENTRY  DETACH_TERMINATED_OWN_TCBS
DSTCB QR    KE 0503 KEDS  EXIT   DETACH_TERMINATED_OWN_TCBS/OK
```
Using Forbidden Functionality

Issue I/O without CICS file control:
• Bypass CICS file control
• “Batch” transactions segregated from normal processing

Disadvantages:
• Cannot issue OPEN/CLOSE in COBOL program
• No backout or forward recovery
• Activity not in dump, trace, etc.
Using Forbidden Functionality

Reminder: the OTE only supports CICS LE service routines:

- COBOL display becomes a WRITEQ TD (not threadsafe until CICS TS 5.2!)
- COBOL dynamic call modified for CICS
- OPEN/CLOSE unavailable
- Storage obtained via EXEC CICS GETMAIN
Segregating Transactions

OTE provides some insulation from difficult transactions

- CPU intensive tasks don’t own QR TCB
- QR available for CEMT, etc.
OTE Performance Considerations

There are several performance issues that are unique to the OTE:

• Non-Threadsafe EXEC CICS commands
• Non-Threadsafe CICS Global User Exits
• Multi-TCB issues with OPENAPI programs
Definitions

Define “threadsafe”

1. “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 does not depend on any area of shared storage remaining consistent between machine instructions.”

2. “A program defined as CONCURRENCY=THREADSAFE is one that will be allowed to run on an open TCB.”

3. “A threadsafe CICS command is one that is allowed to run under an open TCB. A non-threadsafe command is one that is not allowed to run under an open TCB.”
Non-Threadsafe CICS Commands

- Many Some A few commands (still) not Threadsafe
- Use of non-Threadsafe commands is **fully supported** by CICS
- CICS detects non-threadsafe command and switches task to QR TCB
- Task’s TCB status following command depends on API definition
- Potential performance issue for API=OPENAPI
Non-Threadsafe CICS Commands

A list of the commands that are threadsafe can be found in the *CICS Application Programming Reference Manual*, under **CICS threadsafe commands in the API**.

A list of the threadsafe SPI commands can be found in the *CICS System Programming Reference Manual*, in Appendix D, **Threading SPI commands**.
Non-Threadsafe CICS Exits

• Significant area of concern
• Task switched to QR for duration of exit, then back to Open TCB
• Infrequently referenced exits less of a problem
• Frequently referenced exits (eg., XEIIN) are a major performance problem
• XRMIIIN/OUT and Dynamic Plan Selection most worrisome
• Worst case: significant (20%++?) increase in CPU utilization.
• Can cause CPU impact even if FORCEQR=YES
Non-Threadsafe CICS Exits

• Use DFH0STAT to identify exits in use
  – Select DB2, User Exit and Global User Exit options
  – Identifies all active exits by program name, CONCURRENCY option, exit point, and GWA usage
  – Shows Dynamic Plan exits

• Identify vendor exits and contact vendor
  – Do not mark threadsafe without vendor OK
  – Do not convert with heavily used QUASIRENT exits

• Review homegrown exit code to ensure threadsafe
Minimizing CPU Overhead

CPU overhead is incurred when a non-Threadsafe command is issued while the task is running on an Open TCB. Overhead is zero when no non-Threadsafe commands are issued while the task is running on an Open TCB.

EXEC SQL OPEN CURSOR
PERFORM UNTIL ...
    EXEC SQL FETCH....
EXEC CICS WRITEQ TD
END-PERFORM
Minimizing CPU Overhead

Once the command has been identified.....

• Replace it
  Replace Transient Data with CICS TempStor?
• Relocate it
  Move the command outside of the SQL loop?
Minimizing CPU Overhead

Replace Transient Data with CICS Temporary Storage:

EXEC SQL OPEN CURSOR
PERFORM UNTIL ...
   EXEC SQL FETCH....
   EXEC CICS WRITEQ TS
END-PERFORM
Minimizing CPU Overhead

DFH$MOLS of modified program running Threadsafe in test:

EXEC CICS WRITEQ TD replaced with WRITEQ TS

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2REQCT</td>
<td>00004E20</td>
<td>00:00:06.69787</td>
<td>483</td>
</tr>
<tr>
<td>USRDISPT</td>
<td>00066339000001E3</td>
<td>00:00:03.82084</td>
<td>483</td>
</tr>
<tr>
<td>USRCPUT</td>
<td>0003A4D30000001E3</td>
<td>00:00:00.15334</td>
<td>483</td>
</tr>
<tr>
<td>SUSPTIME</td>
<td>000025700000001E3</td>
<td>00:00:00.01558</td>
<td>483</td>
</tr>
<tr>
<td>DISPWTT</td>
<td>000003CE0000001E2</td>
<td>00:00:00.02592</td>
<td>321</td>
</tr>
<tr>
<td>QRD1SPT</td>
<td>00000654000000141</td>
<td>00:00:00.01102</td>
<td>321</td>
</tr>
<tr>
<td>QRCPUT</td>
<td>000002B100000141</td>
<td>00:00:00.01558</td>
<td>482</td>
</tr>
<tr>
<td>KY8DISPT</td>
<td>000659D3000000A1</td>
<td>00:00:06.65937</td>
<td>161</td>
</tr>
<tr>
<td>KY8CPUT</td>
<td>0003A1F7000000A1</td>
<td>00:00:03.80913</td>
<td>161</td>
</tr>
<tr>
<td>L8CPUT</td>
<td>0003A1F7000000A1</td>
<td>00:00:03.80913</td>
<td>161</td>
</tr>
<tr>
<td>QRMODDLY</td>
<td>0000032D000000140</td>
<td>00:00:00.01300</td>
<td>320</td>
</tr>
<tr>
<td>DSCHMDLY</td>
<td>0000033C000000144</td>
<td>00:00:00.01324</td>
<td>324</td>
</tr>
</tbody>
</table>
Minimizing CPU Overhead

QR TCB
Task Starts

FETCH

Open TCB

DB2 Code executes

WRITEQ TS

FETCH

WRITEQ TS

Copyright (c) 2013, The Evans Group, Inc.
Minimizing CPU Overhead

Relocate Transient Data Writes:

EXEC SQL OPEN CURSOR
PERFORM UNTIL ...
  PERFORM VARYING...
    EXEC SQL FETCH...
    MOVE RESULTS TO WS-RESULTS()
  END-PERFORM
PERFORM VARYING...
  EXEC CICS WRITEQ TD FROM(WS-RESULTS())
  END-PERFORM
END-PERFORM
Minimizing CPU Overhead

DFH$MOLS of modified program running Threadsafe in test
Results of 10 SQL FETCH placed in Working Storage, then
issue 10 EXEC CICS WRITEQ TD at once

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2REQCT</td>
<td>00004E20</td>
<td></td>
<td>20000</td>
</tr>
<tr>
<td>USRDISPT</td>
<td>00066339000001E3</td>
<td>00:00:06.69787</td>
<td>2612</td>
</tr>
<tr>
<td>USRCPUT</td>
<td>0003A4D30000001E3</td>
<td>00:00:03.82084</td>
<td>2612</td>
</tr>
<tr>
<td>SUSPTIME</td>
<td>000025700000001E3</td>
<td>00:00:00.15334</td>
<td>2612</td>
</tr>
<tr>
<td>DISPWTT</td>
<td>000003CE0000001E2</td>
<td>00:00:00.01558</td>
<td>2611</td>
</tr>
<tr>
<td>QRDISPT</td>
<td>00000654000000141</td>
<td>00:00:00.02592</td>
<td>1052</td>
</tr>
<tr>
<td>QRCPUT</td>
<td>000002B1000000141</td>
<td>00:00:00.01102</td>
<td>1052</td>
</tr>
<tr>
<td>KY8DISPT</td>
<td>000659D30000000A1</td>
<td>00:00:06.65937</td>
<td>526</td>
</tr>
<tr>
<td>KY8CPUT</td>
<td>0003A1F70000000A1</td>
<td>00:00:03.80913</td>
<td>526</td>
</tr>
<tr>
<td>L8CPUT</td>
<td>0003A1F70000000A1</td>
<td>00:00:03.80913</td>
<td>526</td>
</tr>
<tr>
<td>QRMODDLY</td>
<td>0000032D000000140</td>
<td>00:00:00.13000</td>
<td>1050</td>
</tr>
<tr>
<td>DSCHMDLY</td>
<td>0000033C000000144</td>
<td>00:00:00.01324</td>
<td>1055</td>
</tr>
</tbody>
</table>
Minimize OTE Overhead: Dummy (OPENAPI) TRUE

CPU overhead is minimized when no non-Threadsafecommands are issued between the DMYRMCAL and theend of OTE user code

PERFORM UNTIL ...
  CALL ‘DMYRMCAL’
  [ote user code]
  EXEC CICS WRITEQ TD
END-PERFORM
Minimize OTE Overhead: Dummy TRUE

QR TCB

Task Starts

CALL ‘DMYRMCAL’

OTE user code

WRITEQ TD

CALL ‘DMYRMCAL’

OTE user code

WRITEQ TD

Open TCB
Minimize OTE Overhead: OPENAPI Program

CPU overhead is minimized when:

1. No non-Threadsafe commands are issued by the program

2. If USER key, no DB2 or OPENAPI TRUE calls issued by the program
Minimize OTE Overhead: OPENAPI Program

MQ Series With OPENAPI program in USER key

L9 TCB
Task Starts

EXEC SQL
DB2 code executes
DB2 code complete

E.C. WRITEQ TD
WRITEQ TD starts
WRITEQ TD ends

MQ PUT
MQ code executes
MQ code complete

E.C. RETURN
Task termination

Copyright (c) 2013, The Evans Group, Inc.
Minimize OTE Overhead: OPENAPI Program

MQ Series With OPENAPI program in CICS key

<table>
<thead>
<tr>
<th>L9 TCB</th>
<th>L8 TCB</th>
<th>QR TCB</th>
<th>MQ TCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td></td>
<td>Task Starts</td>
<td></td>
</tr>
<tr>
<td>DMYTRUE executes</td>
<td>CALL ‘DMYRMCAL’</td>
<td>Threadsafe code</td>
<td></td>
</tr>
<tr>
<td>EXEC SQL</td>
<td>WRITEQ TD starts</td>
<td>WRITEQ TD ends</td>
<td>MQ code executes</td>
</tr>
<tr>
<td>E.C. WRITEQ TD</td>
<td>MQ code complete</td>
<td>Task termination</td>
<td></td>
</tr>
</tbody>
</table>

Copyright (c) 2013, The Evans Group, Inc.
Minimize OTE Overhead: REQUIRED Program with API(CICSAPI)

CPU overhead is minimized when:

1. No non-Threadsafe commands are issued by the program
Reducing CPU Overhead

Note:
Prior to CICS 4.2, IRC is not threadsafe. This means that
Threadsafe commands that are function shipped will be
treated as if they are non-threadsafe.
CICS 4.2 IPIC connections support threadsafe mirror
transactions
Ensuring Threadsafe Coding When Creating New Programs

Design is critical

- Ensure threadsafe coding standards are met
- Minimize number of TCB switches
Ensuring Threadsafe Coding When Creating New Programs

Ensure Threadsafe Coding Standards

• Eliminate updates to shared storage areas:
  – CWA
  – GWA
  – GETMAIN(SHARED)
  – OS GETMAIN
  – LOAD HOLD
• Require use of RENT on link-edit step
• Use RENTPGM=PROTECT in CICS
Ensuring Threadsafe Coding When Creating New Programs

Minimize number of TCB switches

- Maximum performance
- Use only Threadsafe commands
- Design program flow to cluster OTE usage
- Issue non-Threadsafe commands before or after OTE activity complete
Diagnosing Threadsafe Problems

No way to prove threadsafe!

- Threadsafe problems most likely to occur during peak time.
- Stress testing more likely to bring out threadsafe problems.
- Best way to ensure success is strong application knowledge.
- Be thorough in your review.
Diagnosing Threadsafe Problems

How to tell when Testing is Complete?

• Errors based on probability
• Difficult to force simultaneous execution of code path
• Use stress testing
  – Set MAXTASK high
  – Set DSALIMITs high
  – Set SYSDDUMPING on!
  – Use driver program to issue large number of STARTs
Diagnosing Threadsafe Problems

Unpredictable Results Means Just That!

- Difficult to identify
- “Impossible” behavior likely to be threadsafe issue
- Use CICS auxtrace
- CICS system dump
Diagnosing Threadsafe Problems

Paired MVS macros that need same TCB

- Macros such as ENQ and DEQ must run on same TCB
- Intervening user code can force TCB switch
- Second macro in pair fails
- Macros include:
  - ENQ/DEQ
  - ATTACH/DETACH
Diagnosing Threadsafe Problems

A Statically Called Assembler Program Isn’t Threadsafe

ASMPGM1  CSECT
       LA R13, SAVEAREA
       STM R14,R12,12(R13)
       .
       LM R14,R12,12(R13)
       BR R14
       .
       .
       SAVEAREA DS 18F
Diagnosing Threadsafe Problems

All Called Routines Run on TCB of the Caller

- Because ASMPGM1 issues no CICS commands, the code runs normally in a non-threadsafe environment
- CICS is not notified for calls
- Simultaneous access to SAVEAREA results in overlay
- Probable S0C4
- Identifiable in test via RENTPGM=PROTECT
Diagnosing Threadsafe Problems

All Called Routines Run on TCB of the Caller

Possible solutions:

1. Convert ASMPGM1 to Command Level
2. Alter COBPGM to pass address of RSA
3. Leave COBPGM non-Threadsafe
4. Convert ASMPGM1 to LE enabled Assembler
Threadsafe File Control

Threadsafe VSAM RLS available with CICS 3.2
Threadsafe **local** VSAM shipped in CICS 3.2 as disabled

New SIT parm:

```
FCQRONLY=[YES | NO]
```

- FCQRONLY=YES forces all file control to run on QR TCB
- FCQRONLY=NO allows threadsafe file control requests to run on L8/L9 TCB

Remote VSAM on non-IPIC connections remains non-threadsafe
Futures

“It is the intention of IBM for future releases of CICS Transaction Server for z/OS to continue to enhance OTE support to enable the ongoing migration of CICS and application code from the QR to open TCBs.”

Threadsafe considerations for CICS
Recommendations

• Convert XRMIIN/OUT and Dynamic Plan Selection exits **before** migrating to a threadsafe capable CICS release
• Convert all frequently used exit programs to threadsafe before converting programs
• Verify that required maintenance is on CICS and vendor products before converting programs to threadsafe
• Review IBM Redbook “Threadingse Considerations for CICS”
• Beware of COBOL dynamic CALLs