

# **CICS and Threadsafe**

## **Exploiting the Open Transaction Environment**

Russ Evans

[russ@evansgroupconsulting.com](mailto: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

Open TCB

Task Starts

EXEC CICS

EXEC SQL



DB2 Code executes

Application Code



DB2 Code completes

EXEC SQL



DB2 Code executes



DB2 Code completes

# OTE and DB2

With Threadsafe

QR TCB

Open TCB

Task Starts

EXEC CICS

EXEC SQL



DB2 Code executes

Application Code

DB2 Code executes

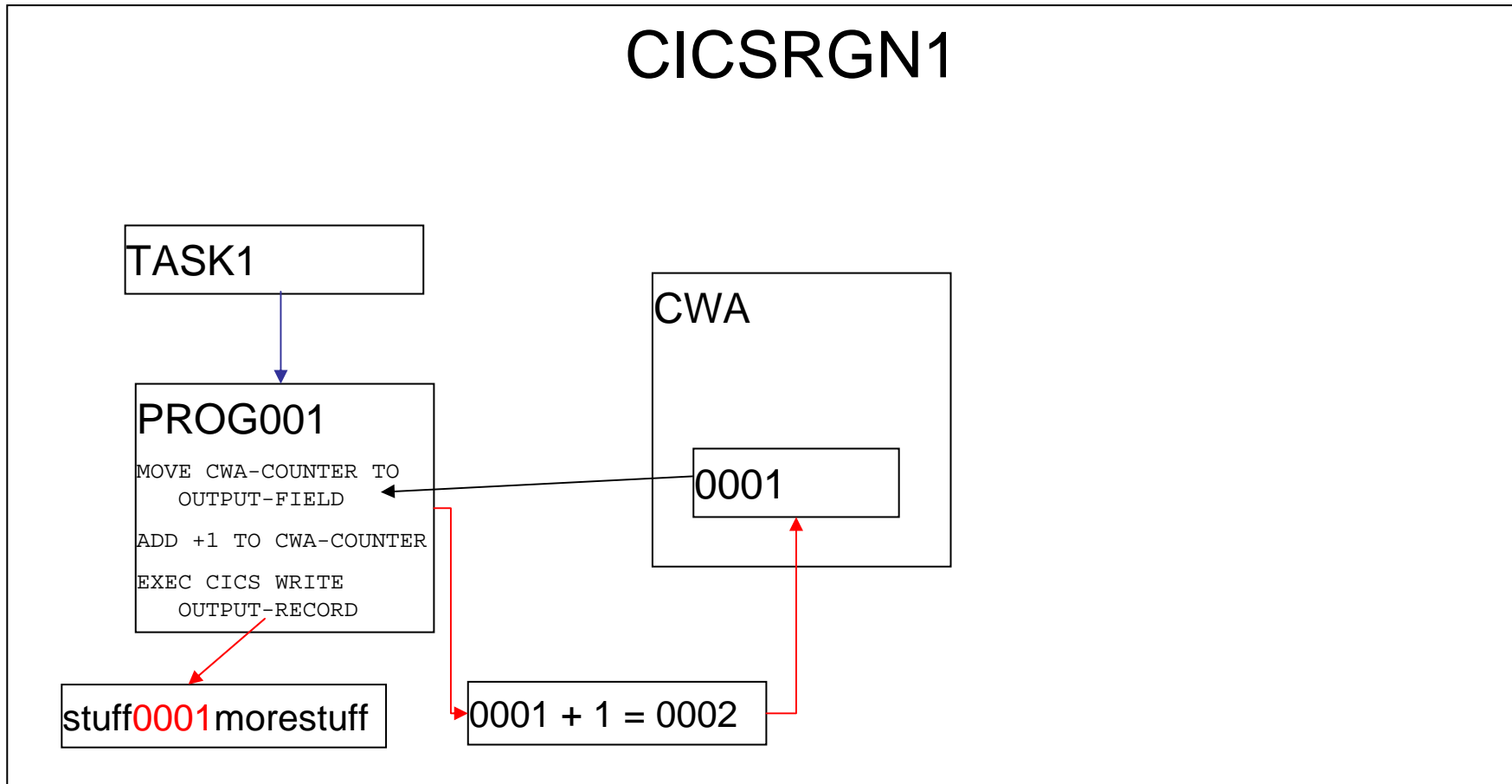
Task Termination



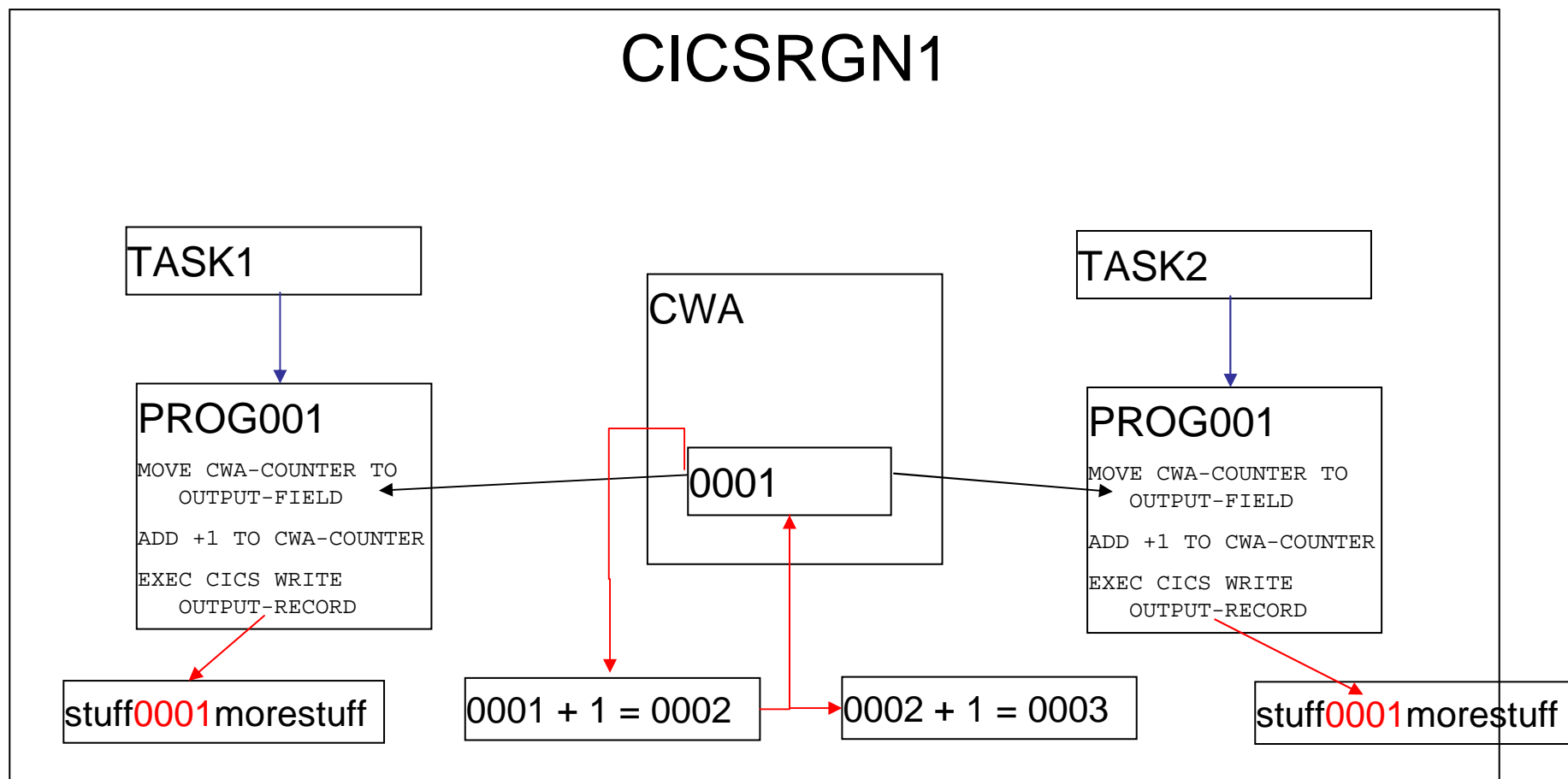
Task completes



# So, What's the Problem



# So, What's the Problem



# 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:
  - New 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 DFHSR0622 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 “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.”



# Identifying Threadsafe Programs

Continued...

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

- Utility DFHEISUP will scan for CICS commands commonly used in non-threadsafe 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

## Making Programs Threadsafe

continued...

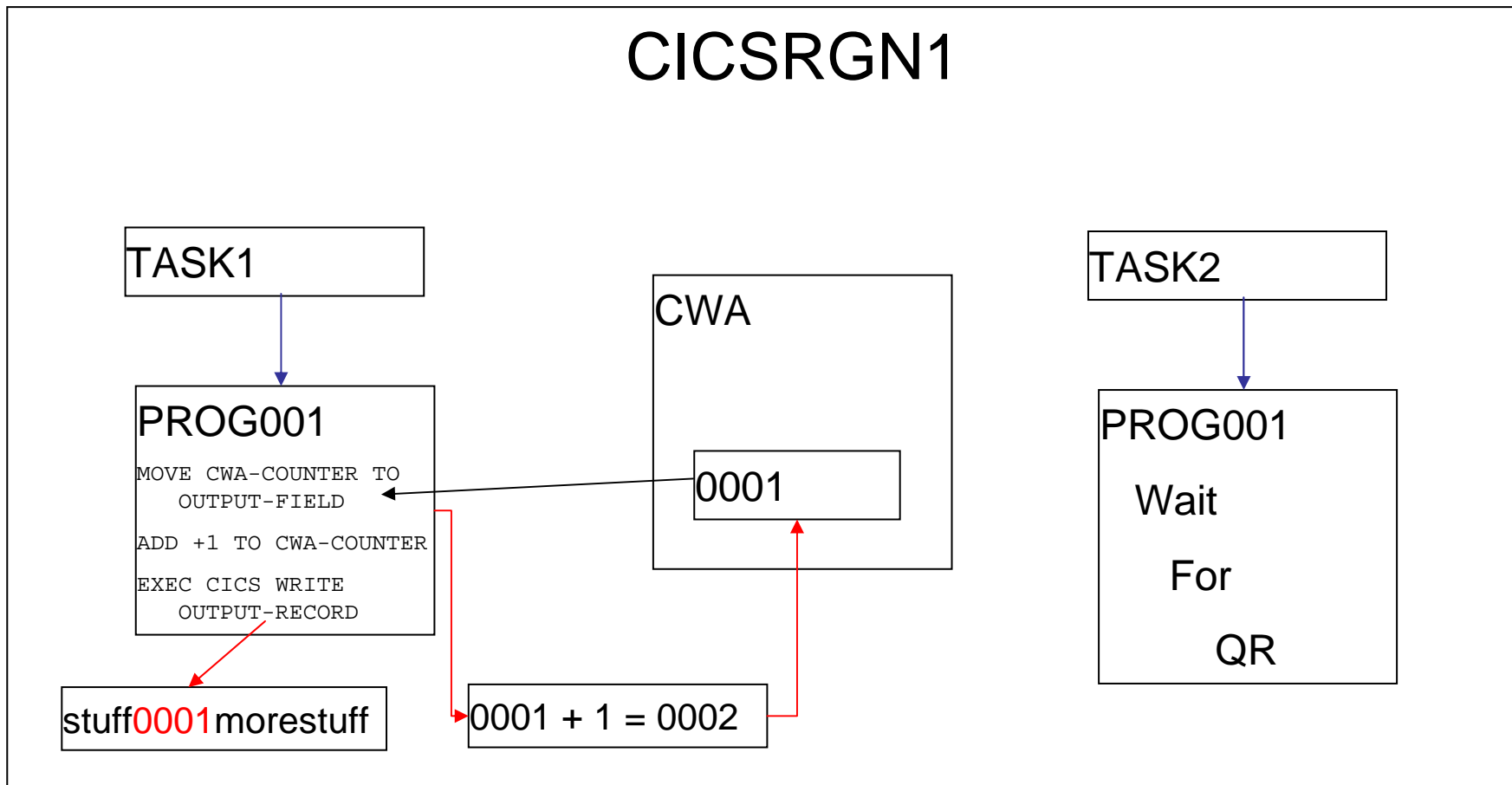
If shared storage use is limited to few programs:

- 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

continued...

Leave Program CONCURRENCY(QUAISRENT)



# Making Programs Threadsafe

continued...

## 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

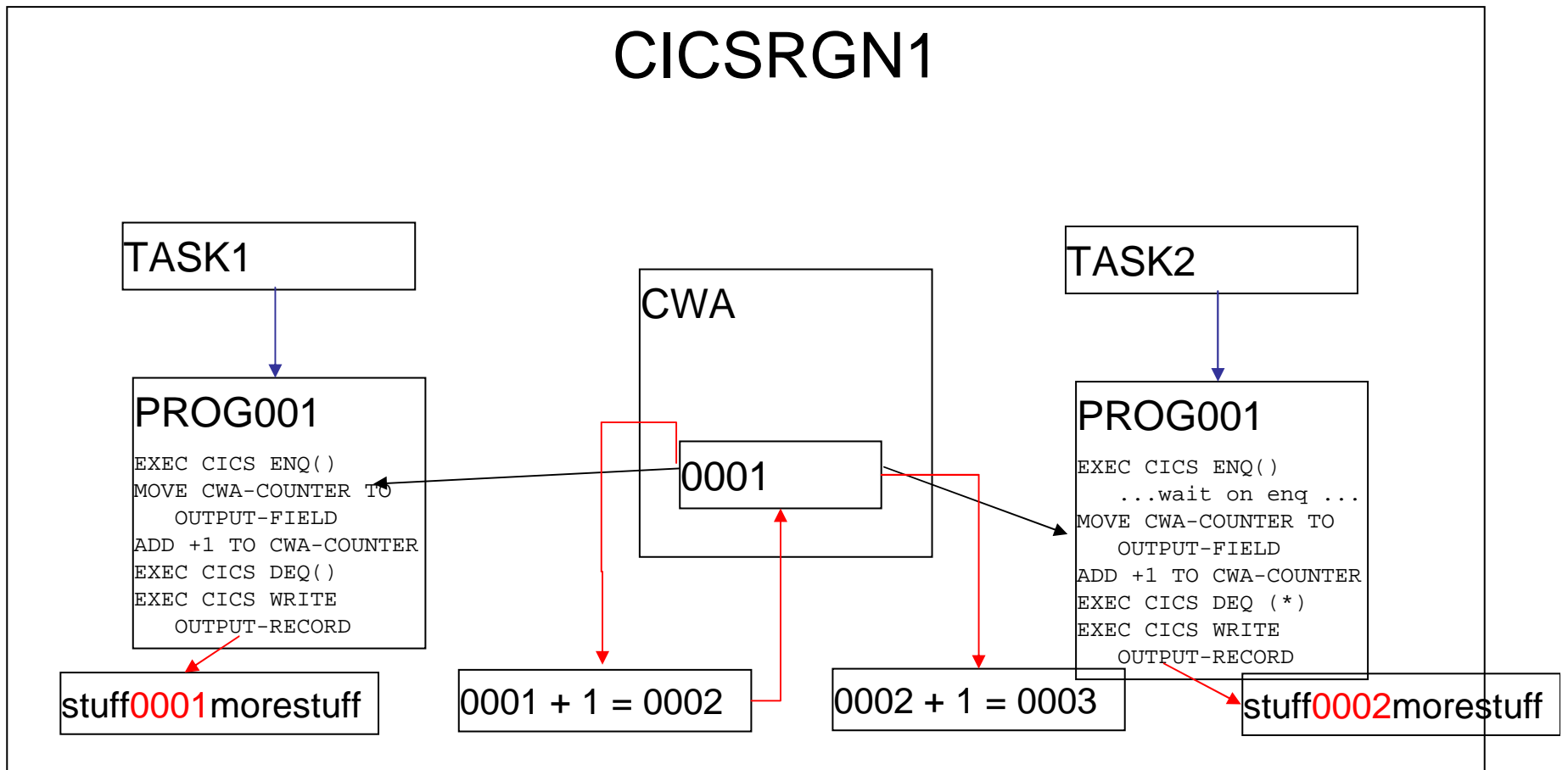
continued...

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

# Making Programs Threadsafe

continued...



# Making Programs Threadsafe

continued...

## ENQ Issues:

- CPU Cost
- Potential bottleneck
  - Limit ENQ duration by issuing DEQ as soon as possible
  - Ensure no possibility of deadly embrace



## Making Programs Threadsafe

continued...

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          Return to caller
                LTORG ,
                END    DMYTRUE
```

# Accessing The OTE

QR TCB

Open TCB

Task Starts

Non-threadsafe code

E.C. non-threadsafe

CALL 'DMYRMCAL' 

DMYTRUE executes

Threadsafe user code

E.C. threadsafe



E.C non-threadsafe

E.C. non-threadsafe

Task Termination

# 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

Open TCB

Task Starts

Non-threadsafe code

E.C. non-threadsafe

CALL 'DMYRMCAL'  DMYTRUE executes

Threadsafe user code

E.C. threadsafe

Non-threadsafe code  CALL 'DMxRMCAL'

Task Termination

## 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 DFHRMCAL 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

## Accessing The OTE CONCURRENCY(REQUIRED)

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

Command Completes



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

# Accessing The OTE

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

# 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:

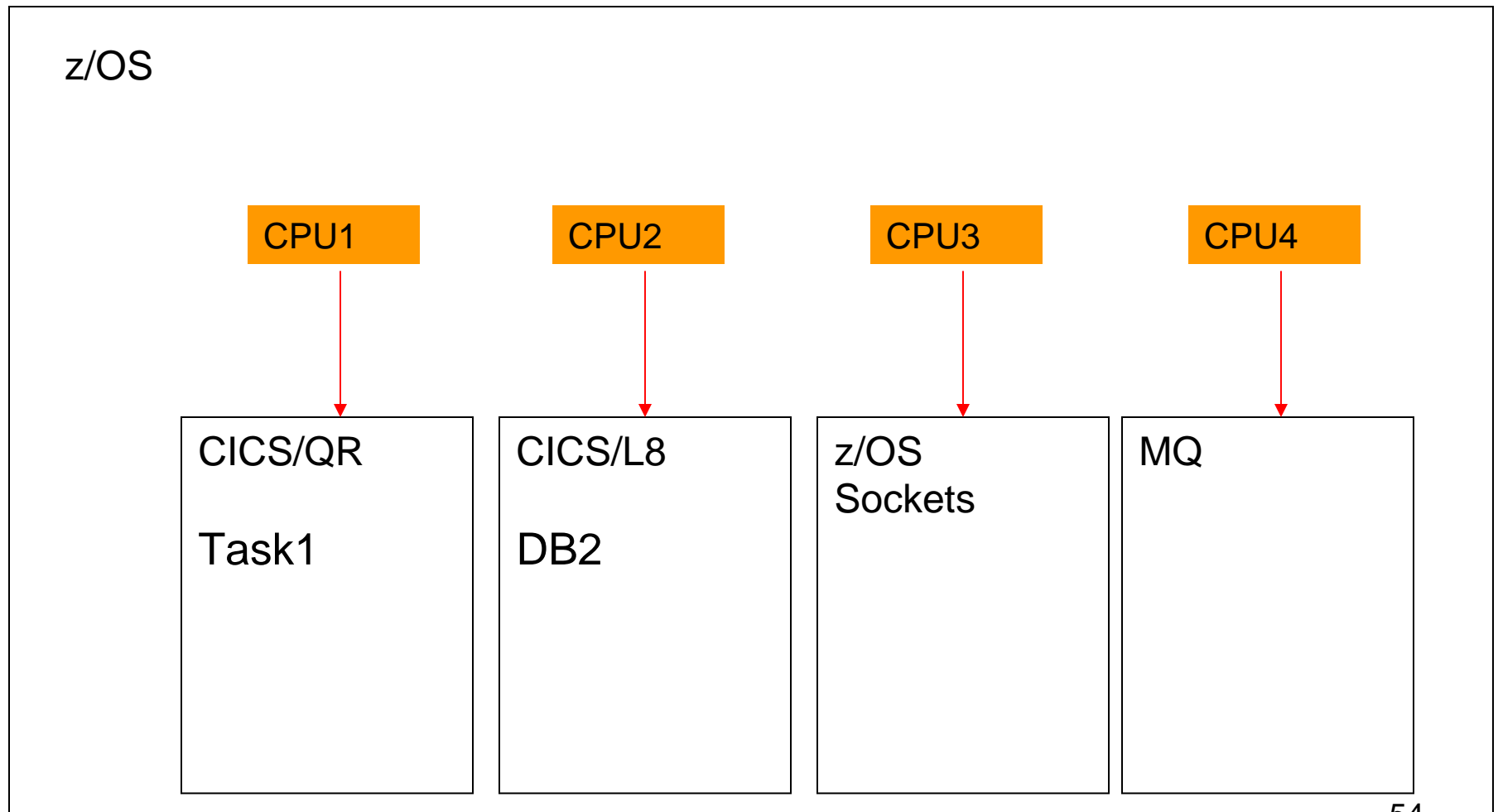
- **Do not attempt to initialize batch LE environment under CICS OPENAPI.**

## 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

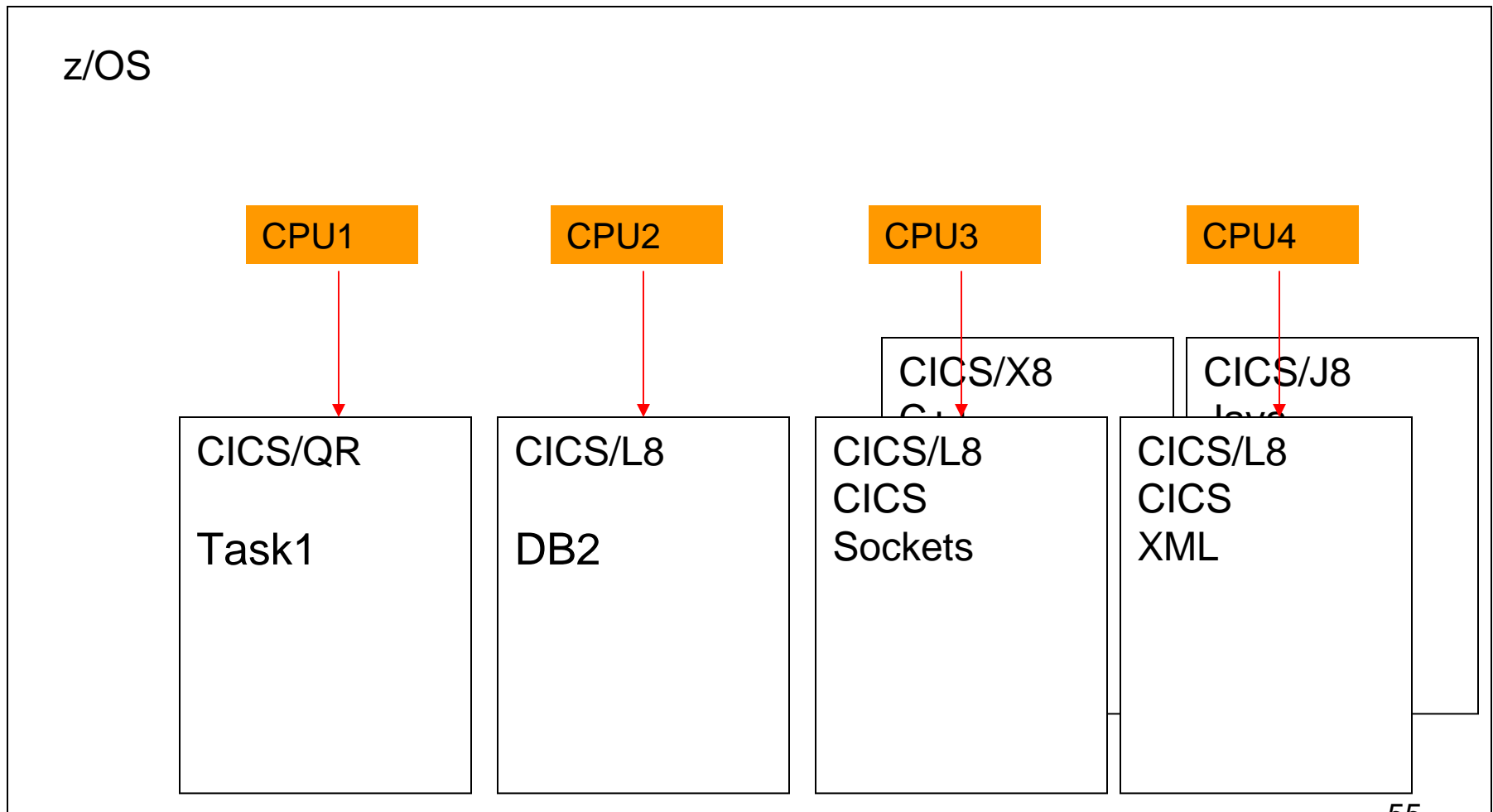
# Multiple TCB Structure

## Classic CICS



# Multiple TCB Structure

Modern CICS



## 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<sup>(\*)</sup>
- 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

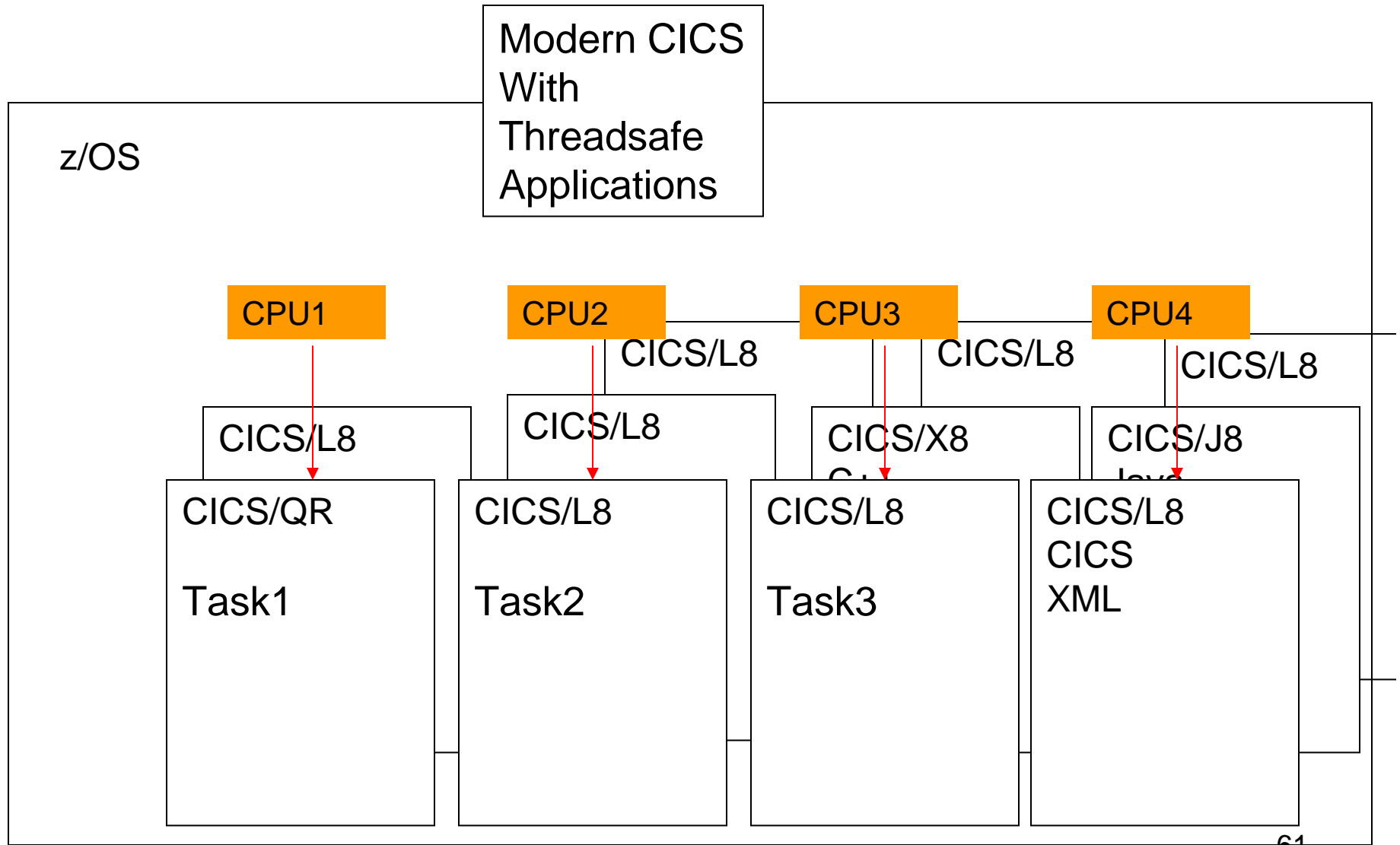
- OTE task “owns” his TCB until next EXEC CICS<sup>(\*)</sup>
- QR is available for other workload
- No region hold-up
- No extended response times
  - Other workload unaffected
  - Response time improves

## 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 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



## 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!)
- COBOL dynamic call modified for CICS
- OPEN/CLOSE unavailable
- Storage obtained via EXEC CICS GETMAIN

# Segregating Transactions

OPE 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 commands 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, **Threadsafe 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., XEIN) are a major performance problem
- XRMIIN/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

## Using IBM Utility DFH\$MOLS

- IBM supplied utility to analyze SMF 110 records
- Provides detailed report
  - One page / task
  - Storage utilization
  - CPU utilization
    - By TCB type
  - Response time
- Can use pre-generated MCT A\$
- Activate monitoring with CEMT
  - SET MON ON PER
- Flush buffers with CEMT
  - SET MON ON NOP

# Using IBM Utility DFH\$MOLS

## Use IFASMFDP to extract the 110 records

INDDx points to your SMF datasets. You can use either active datasets or archives

```
//*****  
*****  
//* Step 1: Unload data from the SMF data sets  
//*****  
*****  
//SMFDUMP EXEC PGM=IFASMFDP  
//INDD1 DD DSN=SYS1.D002.MAN11,DISP=SHR,AMP=( 'BUFSP=65536' )  
//INDD2 DD DSN=SYS1.D002.MAN12,DISP=SHR  
//INDD3 DD DSN=SYS1.D002.MAN13,DISP=SHR  
//OUTDD1 DD DSN=?????.SMF.DATA1,DISP=(NEW,CATLG),  
// SPACE=(CYL,(50,10)),UNIT=SYSDA  
//SYSPRINT DD SYSOUT=A  
//SYSIN DD *  
  
INDD(INDD1,OPTIONS(DUMP))  
INDD(INDD2,OPTIONS(DUMP))  
INDD(INDD3,OPTIONS(DUMP))  
OUTDD(OUTDD1,TYPE(110(1)))
```

OUTDD1 points to the output dataset that holds the extracted 110 records

Use an INDD control statement to describe each SMF file used as input.

The OUTDD control statement describes your output file and the record types to be extracted. We're using 110 subtype 1 records

# Using IBM Utility DFH\$MOLS

Use DFH\$MOLS to format the extracted records

INPUT DD points to  
OUTDD dataset  
from previous step.

```
//PRNT      EXEC PGM=DFH$MOLS
//STEPLIB   DD DSN=SYS2.CICSTS31.CICS.SDFHLOAD,DISP=SHR
//INPUT     DD DSN=?????.SMF.DATA1,DISP=OLD
//SORTWK01  DD SPACE=(CYL,(5,1)),UNIT=SYSDA
//SORTWK02  DD SPACE=(CYL,(5,1)),UNIT=SYSDA
//SORTWK03  DD SPACE=(CYL,(5,1)),UNIT=SYSDA
//SORTWK04  DD SPACE=(CYL,(5,1)),UNIT=SYSDA
//SORTWK05  DD SPACE=(CYL,(5,1)),UNIT=SYSDA
//SORTDIAG  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SYSABEND  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSIN     DD *
SELECT TRANID=trn1,trn2
DATE START=03/23/2006
/*
```

The report is  
written to  
SYSPRINT

Use the SELECT  
TRANID cards to limit  
your report.

Use the DATE START card  
to limit your report

# Using IBM Utility DFH\$MOLS

FIELD-NAME	UNINTERPRETED	INTERPRETED
DFHTASK C001	TRAN C5E2C3F1	ESC1
DFHTERM C002	TERM C3D7F8F4	CP84
DFHCICS C089	USERID C3C9C3E2 C4F2F2F4	CICSD224
DFHTASK C004	TTYPE E3D60000	TO
DFHCICS T005	START BED82B7ADC91D761	2006/05/23 10:53:46.968349
DFHCICS T006	STOP BED82B7ADD3A7B40	2006/05/23 10:53:46.971047
DFHTASK P031	TRANNUM 0000513C	513
DFHTASK A109	TRANPRI 00000001	1
...		
DFHTERM C111	LUNAME E2F0F1E3 C3D7F8F4	S01TCP84
DFHPROG C071	PGMNAME C5E2D7E4 E2C5C3F1	ESPUSEC1
DFHTASK C097	NETUOWPX C2C8C4D5 C5E34BE2 F0F1E3C3 D7F8F400 00000000	BHDNET.S01TCP84
DFHTASK C098	NETUOWSX D82B7ADC9D100001	
DFHCICS A131	PERRECNT 00000001	1
DFHTASK T132	RMUOWID BED82B7ADC9D1021	2006/05/23 10:53:46.968529
DFHCICS C167	SRVCLSNM C3C9C3E2 40404040	CICS
...		
DFHTASK C163	FCTYNAME C3D7F8F4	CP84
DFHTASK A164	TRANFLAG 4000800002000000	
DFHTERM A165	TERMINFO 01000191	
...		
DFHTASK C082	TRNGRPID 180FC2C8C4D5C5E3...	
DFHTERM C197	NETID C2C8C4D5 C5E34040	BHDNET
DFHTERM C198	RLUNAME E2F0F1E3 C3D7F8F4	S01TCP84

## Non-Threadsafe CICS Exits

DFH\$MOLS report of non-threadsafe program:

DB2REQCT		<b>14879</b>
USRCPUT	00:00:01.11961	<b>29763</b>
SUSPTIME	00:00:01.79190	29763
DISPWTT	00:00:01.69950	29762
QRDISPT	00:00:00.37627	14882
QRCPUT	00:00:00.01568	14882
KY8DISPT	00:00:03.67361	14880
KY8CPUT	00:00:01.10212	14880
L8CPUT	00:00:01.10212	14880
RMITIME	00:00:03.37489	14880

## Non-Threadsafe CICS Exits

### DFH\$MOLS report of non-threadsafe EXIT:

DB2REQCT		14879
USRCPUT	00:00:01.15467	59519
SUSPTIME	00:00:02.71036	59519
DISPWTT	00:00:02.41534	59518
QRDISPT	00:00:00.63364	29760
QRCPUT	00:00:00.01456	29760
KY8DISPT	00:00:03.35622	29759
KY8CPUT	00:00:01.14011	29759
L8CPUT	00:00:01.14011	29759
RMITIME	00:00:02.92852	14880



## 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. Overhead is minimized when non-Threadsafe commands can be clustered on the QR

```
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

DB2REQCT	00004E20	<b>20000</b>	
USRDISPT	00066339000001E3	00:00:06.69787	<b>483</b>
USRCPUT	0003A4D3000001E3	00:00:03.82084	483
SUSPTIME	00002570000001E3	00:00:00.15334	483
DISPWTT	000003CE000001E2	00:00:00.01558	482
QRDISPT	0000065400000141	00:00:00.02592	321
QRCPUT	000002B100000141	00:00:00.01102	321
KY8DISPT	000659D3000000A1	00:00:06.65937	161
KY8CPUT	0003A1F7000000A1	00:00:03.80913	161
L8CPUT	0003A1F7000000A1	00:00:03.80913	161
QRMODDLY	0000032D00000140	00:00:00.01300	320
DSCHMDLY	0000033C00000144	00:00:00.01324	324

# Minimizing CPU Overhead

QR TCB

Task Starts

FETCH



Open TCB

DB2 Code executes

WRITEQ TS

FETCH

WRITEQ TS

## 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

DB2REQCT	00004E20	20000	
USRDISPT	00066339000001E3	00:00:06.69787	2612
USRCPUT	0003A4D3000001E3	00:00:03.82084	2612
SUSPTIME	00002570000001E3	00:00:00.15334	2612
DISPWTT	000003CE000001E2	00:00:00.01558	2611
QRDISPT	0000065400000141	00:00:00.02592	1052
QRCPUT	000002B100000141	00:00:00.01102	1052
KY8DISPT	000659D3000000A1	00:00:06.65937	526
KY8CPUT	0003A1F7000000A1	00:00:03.80913	526
L8CPUT	0003A1F7000000A1	00:00:03.80913	526
QRMODDLY	0000032D00000140	00:00:00.01300	1050
DSCHMDLY	0000033C00000144	00:00:00.01324	1055

## Minimize OTE Overhead: Dummy TRUE

CPU overhead is minimized when no non-Threadsafe commands are issued between the DMYRMCAL and the end of OTE user code

```
PERFORM UNTIL ...  
    CALL 'DMYRMCAL'  
    [ote user code]  
    EXEC CICS WRITEQ TD  
END-PERFORM
```



# Minimize OTE Overhead: Dummy TRUE

QR TCB

Open TCB

Task Starts

CALL 'DMYRMCAL' 

OTE user code



WRITEQ TD

CALL 'DMYRMCAL' 

OTE user code



WRITEQ TD

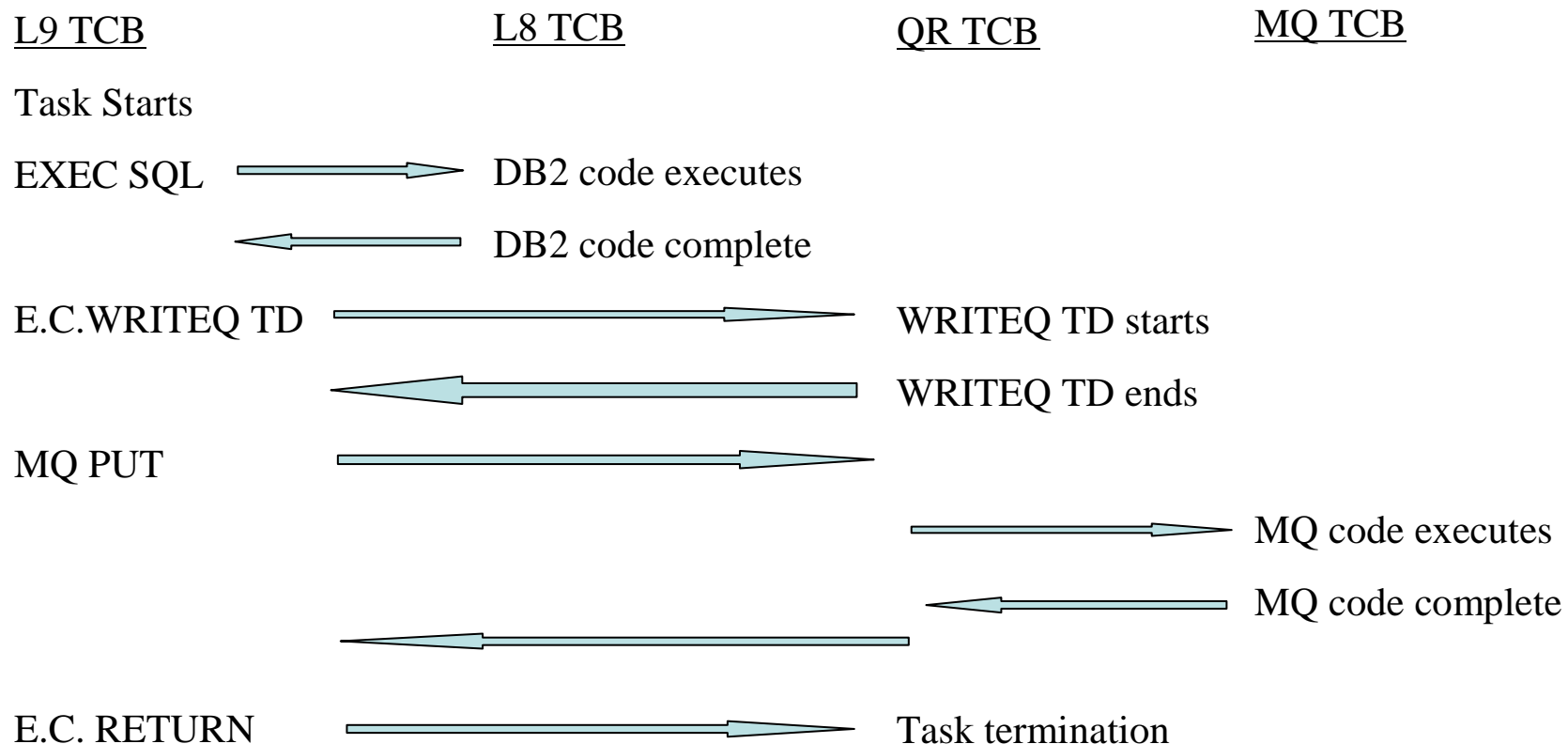
## Minimize OTE Overhead: OPENAPI Program

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

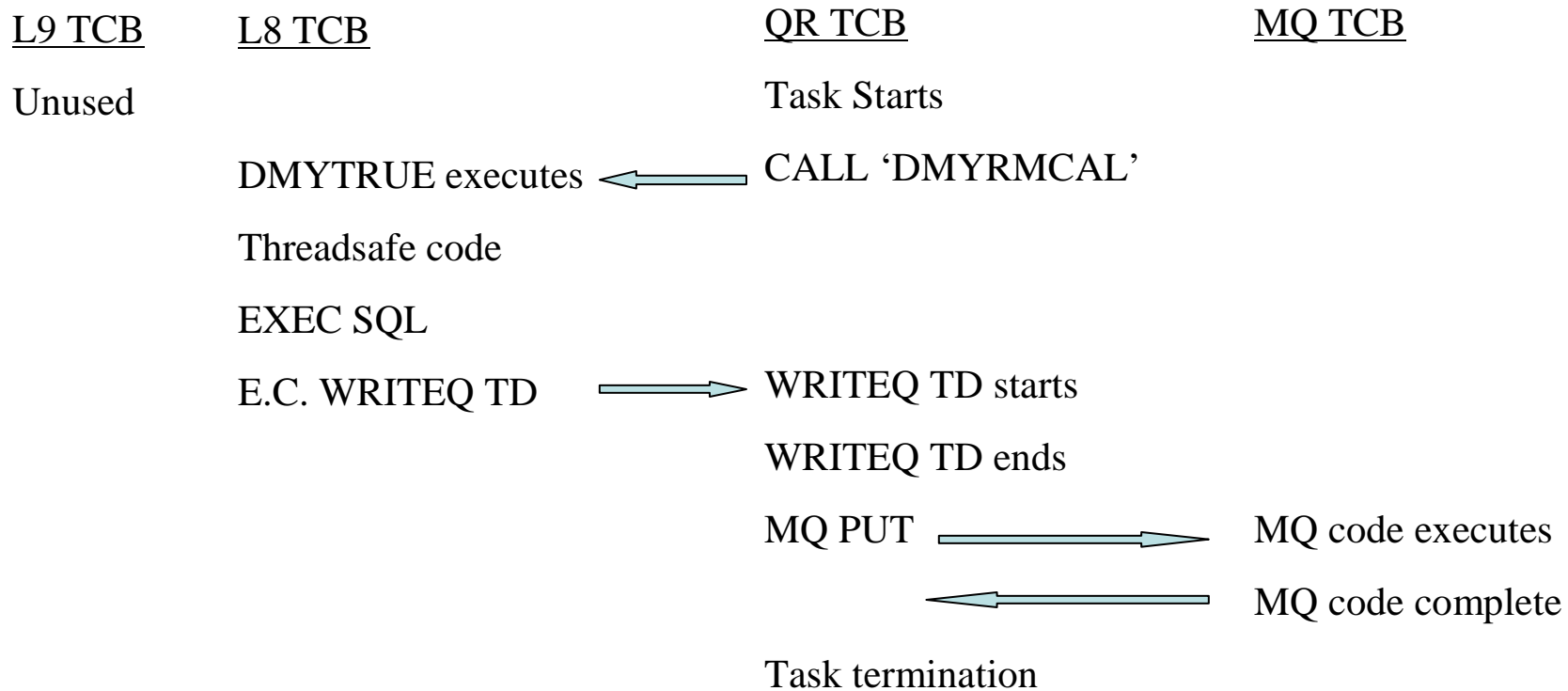
# Minimize OTE Overhead: OPENAPI Program

## MQ Series With OPENAPI program in USER key



# Minimize OTE Overhead: OPENAPI Program

## MQ Series With OPENAPI program in CICS key



# Minimize OTE Overhead: OPENAPI Program

## Relocation Ineffective for OPENAPI!

QR TCB

Open TCB

Task Starts

OTE user code

WRITEQ TS

Inner Loop



WRITEQ TD



WRITEQ TD



WRITEQ TD

Outer Loop

# Minimize OTE Overhead: REQUIRED Program with API(CICSAPI)

CPU overhead is minimized when:

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

# Minimize OTE Overhead: REQUIRED Program Relocation Ineffective for REQUIRED!

QR TCB

Open TCB

Task Starts

OTE user code

WRITEQ TS

Inner Loop



WRITEQ TD



WRITEQ TD



WRITEQ TD

Outer Loop

## 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 SYSDUMPING 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
- Use homegrown application trace
- 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

```
COBPGM  
CALL 'ASMPGM1'  
USING PARM-LIST.
```

```
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

## Threadsafe File Control

Enable local VSAM threadsafe in CICS 3.2 with PTF  
UK37688

VSAM APARs OA20352 and OA24071 are required

**NOTE: UK37688 changes the default on FCQRONLY from NO to YES. If you are running VSAM RLS threadsafe, and take the default on FCQRONLY, applying UK376688 will disable RLS 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

## Futures

- IBM committed to making more commands threadsafe
- IBM Announces additional threadsafe commands in every release since TS 2.2
- CICS 3.2 introduces threadsafe file control (local)  
Note, CICS TS 3.2 was shipped with threadsafe VSAM disabled. Apply PK45354 to activate it
- CICS 4.2 introduced threadsafe DBCTL for DLI
- Conversion to OPENAPI TRUEs for CICS Sockets, MQ
- Internal use of OPENAPI for CPU intensive processes

## Recommendations

- Consider Threadsafe implications now.
- Heavy CPU users exploit multiprocessors
- Don't forget purchased packages
- Beware of COBOL calls (dynamic or static)

## 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 “Threadsafe Considerations for CICS”