IBM

CICS Threadsafe Whitepaper

The ability to run a CICS transaction as *threadsafe,* or under the auspices of the CICS Open Transaction Environment (OTE), first came to the fore with CICS Transaction Server Version 1.3. Since then, the CICS development team have been engineering as much code as possible to run as threadsafe, exploiting the parallelism of the z/Series processors today. The following extract from the Threadsafe Redbook (see appendix) describes this in detail:

The CICS Open Transaction Environment (OTE)

CICS Open Transaction Environment (OTE) is an architecture that was introduced mainly for three purposes:

- *To increase throughput via more concurrency*
- *To improve performance*
- *To introduce the possibility to use non-CICS APIs*

Prior to OTE, all application code ran under the main CICS TCB called the Quasi-reentrant (QR) TCB (except for some specific VSAM execution, and other specialized activity such as FEPI, security calls, and file opens and closes, which used other TCBs). The CICS dispatcher sub-dispatches the use of the QR TCB between the different CICS tasks. Each task voluntarily gives up control when it issues a CICS service, which then can cause a CICS dispatcher wait. Only one CICS task can be active at any one time on the QR TCB.

But the one and only QR TCB could only execute on one CPU, so CICS execution was only using one physical CPU at a time. For that reason the limit of a specific CICS system's execution capacity was set by the MIPS size of the single CPU of the related MVS system.

SQL calls were done on attached TCBs to prohibit blocking of the QR TCB when a CICS program was waiting for a conclusion of a DB2 request. This feature was called the CICS/DB2 attachment facility.

CICS Transaction Server Version 3 has now expanded OTE usage to not only those applications making DB2 calls, but to any application by means of a new keyword on the program definition.

OTE introduces a new class of TCB, which can be used by applications, called an open TCB. An open TCB is characterized by the fact it is assigned to a CICS task for its sole use, and multiple OTE TCBs can run concurrently in CICS. There is no sub-dispatching of other CICS tasks on an open TCB.

The OTE introduces a lot of new engines (TCBs) to CICS program execution. Each new TCB can execute on one CPU in parallel (concurrently). This gives the potential of increased throughput for a single CICS system, as long as the necessary CPU power is present.

Improved performance

Each new TCB represents a thread where a CICS program can execute in parallel. When the CICS program continues to execute on the open TCB, it is called a threadsafe execution of the program. The result is a reduced number of TCB switches between the open TCB and the QR TCB. This, in turn, results in reduced CPU consumption corresponding to the number of saved TCB switches. The more CICS commands that are made threadsafe the more probability you have of remaining executing on the open TCB.

Quasi-reentrant and threadsafe programs

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 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 given point in time. Compare this with a situation in which multiple instances of the same program are each executing under a separate TCB. In this scenario, multiple tasks would be active in the same program at the same time and the program would have to be fully MVS re-entrant at the very least. For a program to be threadsafe, it must go beyond being fully reentrant and use appropriate serialization techniques when accessing shared resources.

The story so far

So where are we today, with the current release of CICS Transaction Server V3.2, and what does this mean for the majority of customers running their applications who want to take advantage of the CPU savings that are possibly on offer ?

The following diagram plots the progress of OTE through the CICS releases:

Before CICS TS V1.3:
- Direct invocation of non-CICS services not supported
- All user tasks run under the QR TCB
- DB2 adaptor uses subtask TCBs to access DB2
- All DB2 calls incur 2 TCB switches

CICS TS 1.3:
- OTE introduced; exploited by JVMs and Java Hotpooling only
- Non-Java user tasks run under QR TCB
- Subset of CICS API and SPI commands are threadsafe
- DB2 adaptor uses subtask TCBs to access DB2
- All DB2 calls incur 2 TCB switches

CICS TS 2.2:
- TRUEs can exploit OTE; DB2 adaptor first TRUE to invoke OTE
- Number of threadsafe API commands increased
- Threadsafe DB2 applications can execute on L8 TCBs
- Number of TCB switches per task depends on application code

CICS TS 2.3:
- Number of threadsafe API commands increased
- User key JVMs supported
- Multiple JVMs per task supported

CICS TS 3.1:
- Full OTE Extension
- Full application use of open TCBs : OPENAPI using L8 or L9
- C and C++ Programs can use XPLINK and run under X8 or X9
- Number of threadsafe API commands increased

CICS TS 3.2:
- WMQ interface with CICS now threadsafe
- System Autoinstalled GLUEs can be defined as threadsafe
- Number of threadsafe API commands, including local VSAM and RLS, increased

When would I want to consider a threadsafe migration ?

So we understand that by taking advantage of OTE, we can achieve CPU savings, as each TCB switch that takes place uses about 2k instructions to execute. This will reduce the amount of processor time that is needed to complete a transaction because the wait associated with the QR TCB has been reduced, but does not necessarily mean that our transactions can execute any faster. They will, however, be able to handle increased workloads without impacting the overall CPU consumption dramatically.

So what conditions usually occur for us to consider implementing threadsafe applications ?

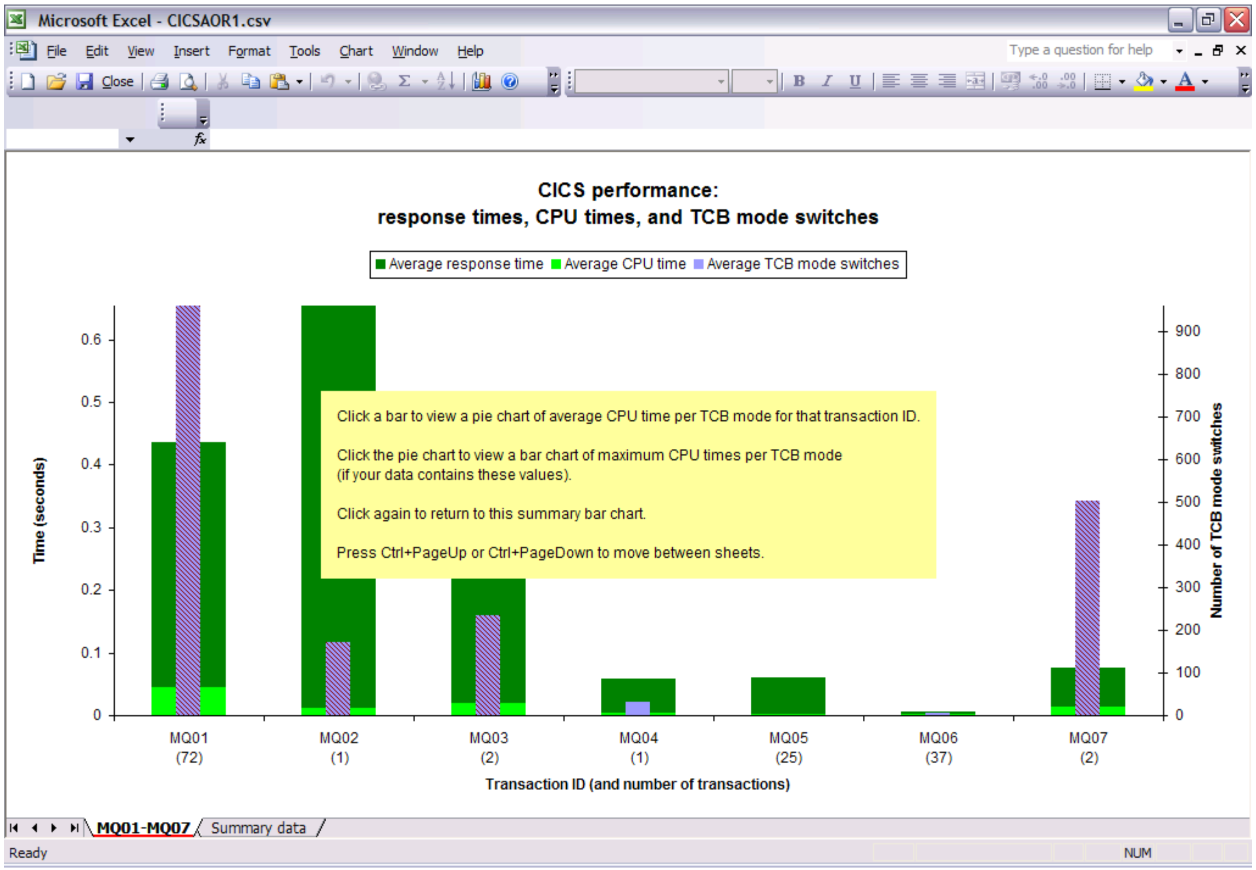
These are some of them:

1. CICS QR TCB is CPU constrained, or running at 100%. This means that every task running on the QR TCB will suffer degradation
2. Applications are competing for the QR TCB. Each task is incurring a re-dispatch delay due to waiting for the QR TCB to become available
3. CICS region itself has become CPU constrained

The benefits of OTE can be summarized as follows:

- **Running on 'Open TCBs' gives CPU concurrency**
- **Throughput is not limited to the speed of a single CP**
- **Reduces any contention for the 'QR' TCB**
- **Open API is not intended to reduce CPU per transaction**
- **It does give more potential throughput per region**
- **Allows the application to use other APIs**
  - Those that would otherwise 'block' the QR TCB

Performance reports will indicate which, if any, of these conditions are prevalent in an installation. Additionally, performance reports can indicate to us which transactions are the best candidates for threadsafe consideration. The following spreadsheet is an extract file produced by CICS Performance Analyzer for z/OS, summarizing transactions by response time. CPU time, and the number of TCB switches that take place:



Click a bar to view a pie chart of average CPU time per TCB mode for that transaction ID.

Click the pie chart to view a bar chart of maximum CPU times per TCB mode (if your data contains these values).

Click again to return to this summary bar chart.

Press Ctrl+PageUp or Ctrl+PageDown to move between sheets.

So at first sight, transactions MQ01 and MQ07 would be our best choice, but it's not as simple as that. We need to consider the characteristics of each individual application, but more importantly, we need to understand the risks associated with undertaking such a migration.

You can create this type of data yourself if you are a current user of CICS TS Version 3, CICS Performance Analyzer for z/OS, and you download Support Pac CP12 from the CICS Tools website:

http://www-01.ibm.com/support/docview.wss?rs=1087&context=SSPPU4&dc=D400&uid=swg24011321&loc=en_US&cs=UTF-8&lang=en

So where's the catch ?

As with all benefits, there are conditions that have to be met before we can proceed. The phrase we like to use is "there's no such thing as a free lunch".

The most suitable application to be considered as a threadsafe candidate can be described as the following:

1. The transaction executes a significant number of SQL calls to DB2
2. All programs that are executed can be defined as threadsafe
3. All exits invoked as part of an SQL call can be defined as threadsafe, and any EXEC CICS commands they execute are threadsafe
4. All exits and EXEC CICS commands executed during the lifetime of the program can be defined as threadsafe

When we define transactions as threadsafe, **we must ensure that the appropriate serialization techniques are used to ensure the integrity of the data** that the application processes. Prior to this, the application relied on the QR TCB and the quasi-reentrant nature of the programs to ensure that data was accessed in the correct manner. Now, we are introducing the ability for a single transaction to run multiple times concurrently, and any shared resources, such as data and storage, are considered to be volatile. It may be that we do not see the results of such a migration until well after the event, so thorough preparation is essential. Customers who have failed to guarantee the integrity of their applications will, at best, suffer transaction failure (best because transaction backout will attempt to preserve data integrity), and at worst, only discover the results when they find corruption in business critical data.

What types of program are considered to be "at risk" for threadsafe ? It's not easy to be specific, but some of the commands you should be looking for are as follows:

- Are my programs using shared resources?
  - CWA
  - Global user exit global work areas
  - Storage acquired explicitly by the application program with the GETMAIN SHARED option
  - Data only Load module

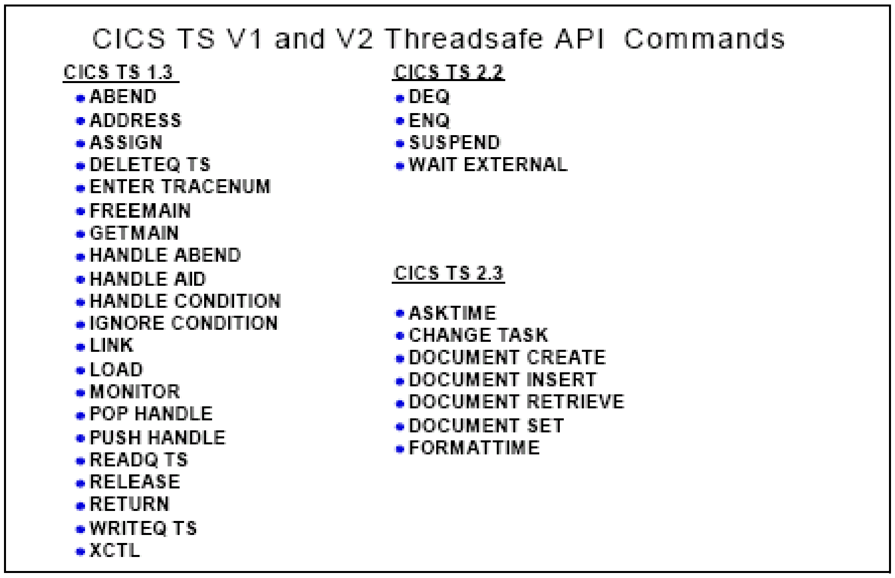
- Which non threadsafe CICS commands is my program using?
  - Is not a data integrity issue but can cause lots of TCB switching

- Which TCB did my CICS commands run on ?

Note: The CICS Threadsafe Redbook includes an excellent example of this when accessing data stored in the CWA

So what is a threadsafe command ?

The following tables indicate those API and SPI commands that have been declared threadsafe, and the CICS TS release level they achieved that:



CICS TS V1 and V2 Threadsafe API Commands

CICS TS 1.3

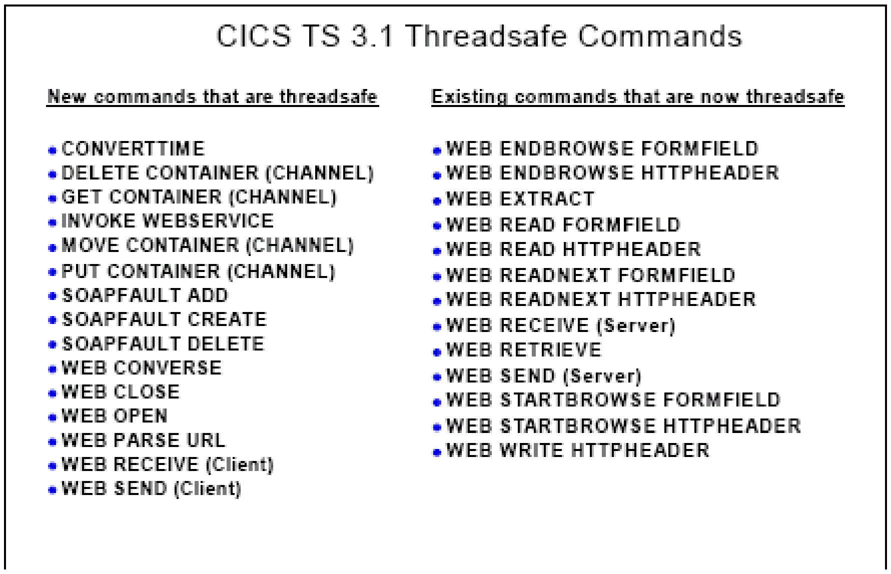
- ABEND
- ADDRESS
- ASSIGN
- DELETEQ TS
- ENTER TRACENUM
- FREEMAIN
- GETMAIN
- HANDLE ABEND
- HANDLE AID
- HANDLE CONDITION
- IGNORE CONDITION
- LINK
- LOAD
- MONITOR
- POP HANDLE
- PUSH HANDLE
- READQ TS
- RELEASE
- RETURN
- WRITEQ TS
- XCTL

CICS TS 2.2

- DEQ
- ENQ
- SUSPEND
- WAIT EXTERNAL

CICS TS 2.3

- ASKTIME
- CHANGE TASK
- DOCUMENT CREATE
- DOCUMENT INSERT
- DOCUMENT RETRIEVE
- DOCUMENT SET
- FORMATTIME



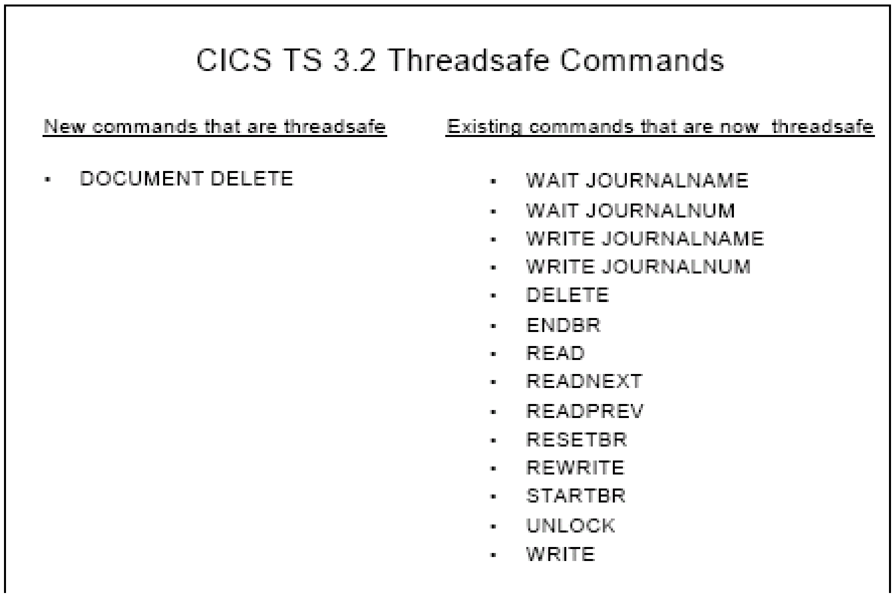
CICS TS 3.1 Threadsafe Commands

New commands that are threadsafe

- CONVERTTIME
- DELETE CONTAINER (CHANNEL)
- GET CONTAINER (CHANNEL)
- INVOKE WEBSERVICE
- MOVE CONTAINER (CHANNEL)
- PUT CONTAINER (CHANNEL)
- SOAPFAULT ADD
- SOAPFAULT CREATE
- SOAPFAULT DELETE
- WEB CONVERSE
- WEB CLOSE
- WEB OPEN
- WEB PARSE URL
- WEB RECEIVE (Client)
- WEB SEND (Client)

Existing commands that are now threadsafe

- WEB ENDBROWSE FORMFIELD
- WEB ENDBROWSE HTTPHEADER
- WEB EXTRACT
- WEB READ FORMFIELD
- WEB READ HTTPHEADER
- WEB READNEXT FORMFIELD
- WEB READNEXT HTTPHEADER
- WEB RECEIVE (Server)
- WEB RETRIEVE
- WEB SEND (Server)
- WEB STARTBROWSE FORMFIELD
- WEB STARTBROWSE HTTPHEADER
- WEB WRITE HTTPHEADER



CICS TS 3.2 Threadsafe Commands

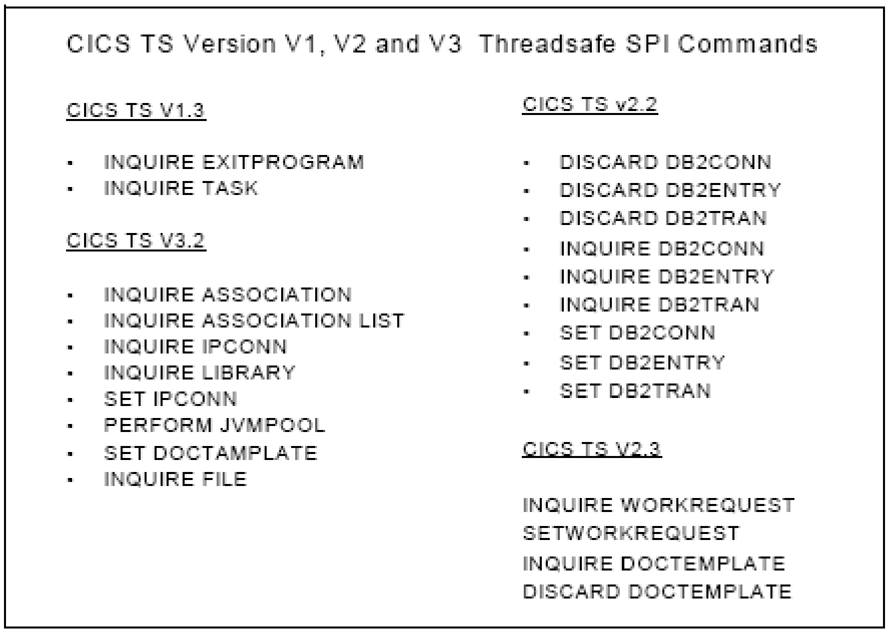
New commands that are threadsafe

- DOCUMENT DELETE

Existing commands that are now threadsafe

- WAIT JOURNALNAME
- WAIT JOURNALNUM
- WRITE JOURNALNAME
- WRITE JOURNALNUM
- DELETE
- ENDBR
- READ
- READNEXT
- READPREV
- RESETBR
- REWRITE
- STARTBR
- UNLOCK
- WRITE

Threadsafe SPI commands



CICS TS Version V1, V2 and V3 Threadsafe SPI Commands

CICS TS V1.3

- INQUIRE EXITPROGRAM
- INQUIRE TASK

CICS TS V3.2

- INQUIRE ASSOCIATION
- INQUIRE ASSOCIATION LIST
- INQUIRE IPCONN
- INQUIRE LIBRARY
- SET IPCONN
- PERFORM JVMPOOL
- SET DOCTAMPLATE
- INQUIRE FILE

CICS TS v2.2

- DISCARD DB2CONN
- DISCARD DB2ENTRY
- DISCARD DB2TRAN
- INQUIRE DB2CONN
- INQUIRE DB2ENTRY
- INQUIRE DB2TRAN
- SET DB2CONN
- SET DB2ENTRY
- SET DB2TRAN

CICS TS V2.3

INQUIRE WORKREQUEST
SETWORKREQUEST
INQUIRE DOCTEMPLATE
DISCARD DOCTEMPLATE

So what causes TCB switching (what's not threadsafe in my program?)

There's no "silver bullet" for making an application threadsafe. In reality, if you refer to the previous spreadsheet, transaction MQ01 is our prime candidate, but this still only executes just over 900 TCB switches on average. If you have transactions that are currently accumulating large numbers of switches, then you could achieve considerable savings, however, be aware, **"your mileage will vary".**

The current status of TCB mode execution with CICS TS V3.2 is this:

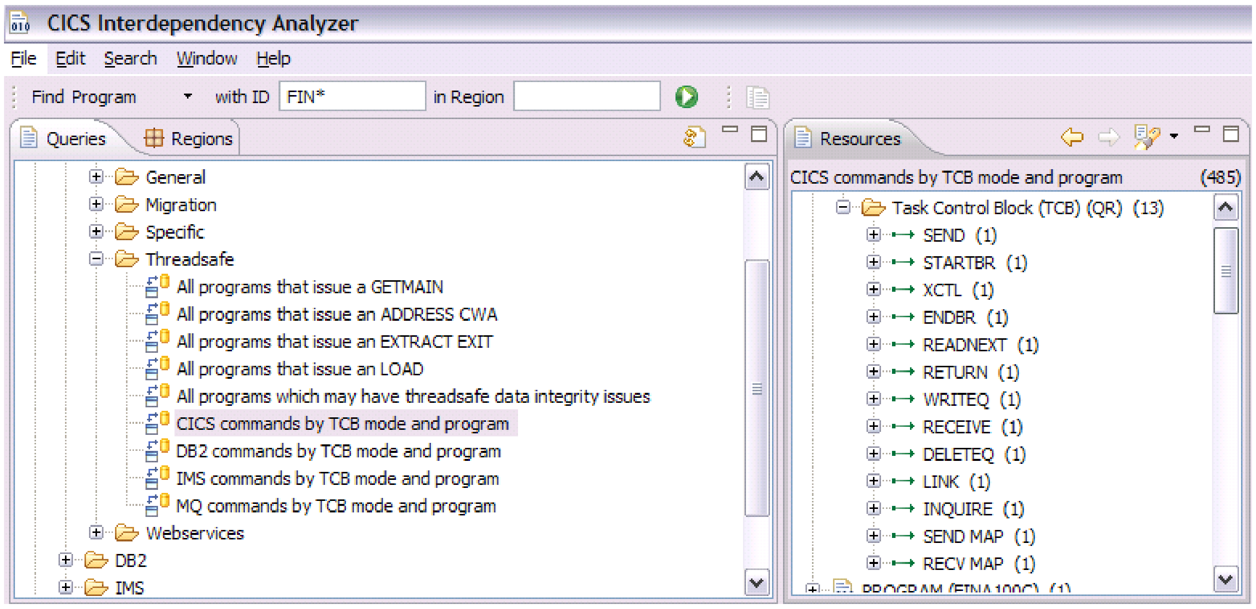
STGPROT	Execkey	Concur	Api	Initial TCB	DB2 command	MQ command	FC command	non-threadsafe command
-----	-----	Quasi	Cics	QR	QR/L8/QR	QR/L8/QR	QR	QR
No	-----	Threadsafe	Cics	QR	L8	L8	no change	QR
Yes	User	Threadsafe	Cics	QR	L8	L8	no change	QR
Yes	Cics	Threadsafe	Cics	QR	L8	L8	no change	QR
No	-----	Threadsafe	Open	L8	L8	L8	L8	L8/QR/L8
Yes	User	Threadsafe	Open	L9	L9/L8/L9	L9/L8/L9	L9	L9/QR/L9
Yes	Cics	Threadsafe	Open	L8	L8	L8	L8	L8/QR/L8

The rules of programs and applications that can run fully under OTE are as follows:

1. Ensure CICS programs are written to the latest standards, as documented in the *CICS Application Programming Guide (SC34-6433)*, for CICS Transaction Server V3.
    a. Compile and link them as reentrant, and reside in read-only storage (SIT parameter RENTPGM=PROTECT)
    b. Use only published CICS interfaces to external resources
2. Use of the Common Work Area (CWA) should be avoided if possible, unless it is for read-only access.
3. All programs, including PLT, User Exits and User Replaceable Modules, should not create or access shared storage (e.g., GETMAIN SHARED)
4. Try to avoid the use of Global Work Areas (GWA) in User Exits
5. All programs and User Exits should only use EXEC CICS threadsafe commands
6. Ensure all programs that are considered to be threadsafe are defined with the CONCURRENCY(THREADSAFE) attribute on the program definition
7. Review function shipping within the application. Any use of this automatically becomes non-threadsafe
8. Keep your CICS regions and applications up to current service levels. Some threadsafe capabilities are delivered through the service channel
9. Check with your independent software vendors for threadsafe capability. Use of non-threadsafe exits will bring a CICS region down
10. For any external resources that require serialization, use the ENQ/DEQ mechanisms

To discover which applications currently fit the threadsafe mould, there are several methods that you can try:

1. CICS Interdependency Analyzer for z/OS
   This IBM product will analyse the API and SPI calls made by your programs, and report on the threadsafe capabilities of each call, including which TCB the call is being made on



- General
- Migration
- Specific
- Threadsafe
  - All programs that issue a GETMAIN
  - All programs that issue an ADDRESS CWA
  - All programs that issue an EXTRACT EXIT
  - All programs that issue an LOAD
  - All programs which may have threadsafe data integrity issues
  - CICS commands by TCB mode and program
  - DB2 commands by TCB mode and program
  - IMS commands by TCB mode and program
  - MQ commands by TCB mode and program
- Webservices
- DB2
- IMS

CICS commands by TCB mode and program (485)

- Task Control Block (TCB) (QR) (13)
  - SEND (1)
  - STARTBR (1)
  - XCTL (1)
  - ENDBR (1)
  - READNEXT (1)
  - RETURN (1)
  - WRITEQ (1)
  - RECEIVE (1)
  - DELETEQ (1)
  - LINK (1)
  - INQUIRE (1)
  - SEND MAP (1)
  - RECV MAP (1)
- PROGRAM (FINA100C) (1)

Figure 1. Screenshot from CICS IA V2.2 showing TCB usage in each program

2. DFHEISUP
   This Load Module scanner will identify those programs that have CICS API and SPI commands from running through the load library. You will need to possibly investigate the program further. This can be run in Summary or Detail mode

3. Running a trace
   If you still have some TCB switching, run a transaction trace to see which command is causing the switch

For more information

- **Redbook: Threadsafe considerations for CICS - SG24-6351-00**
    - **ibm.com**/redbooks

- IBM CICS Transaction Server for z/OS V3.2
    - IBM Software Announcement 204-285, Transaction Servers & Tools e-newsletter, education, and services
      ibm.com/cics

    - Release Guide, Migration Guide, Books and Manuals, Brochures, Demos, and Technical documents
      ibm.com/cics/library/

- IBM CICS Tools
  ibm.com/cics/tools

- IBM z/OS
  ibm.com/servers/eserver/zseries/zos