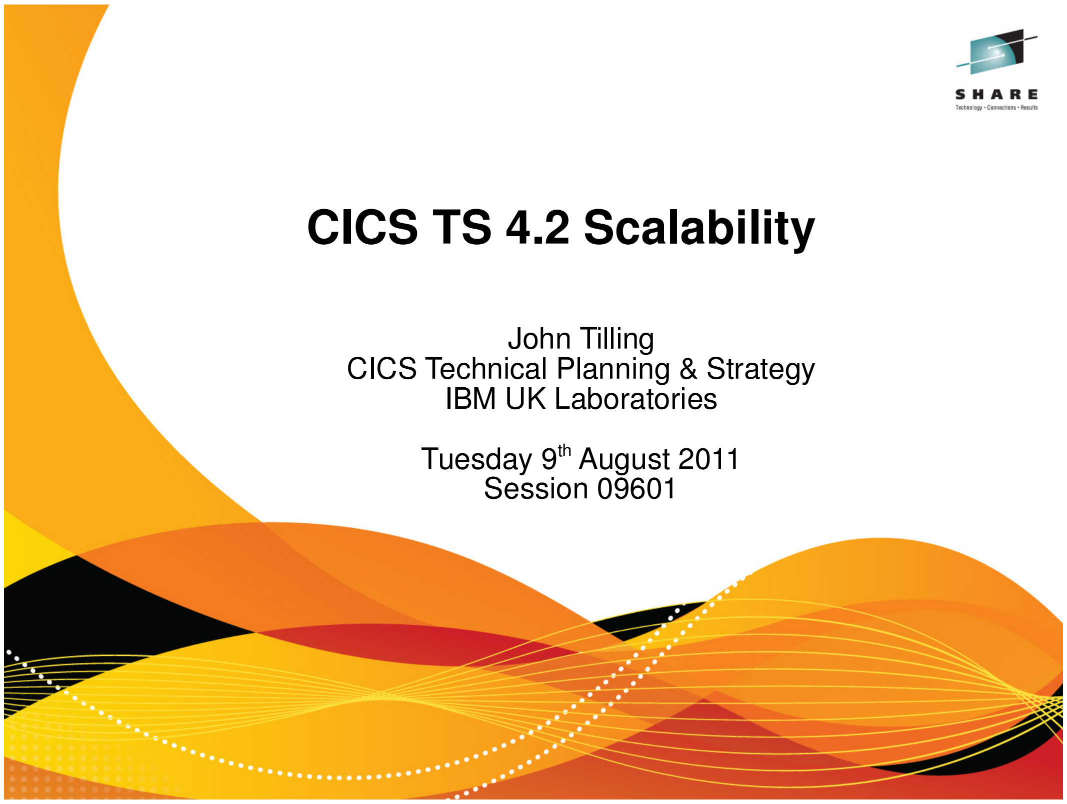


CICS TS 4.2 Scalability

John Tilling
CICS Technical Planning & Strategy
IBM UK Laboratories

Tuesday 9th August 2011
Session 09601

Notes

- The Scalability theme of CICS TS 4.2 is concerned with being able to do more work, more quickly in a single region.

Agenda

- Quick summary of previous release scalability items

- CICS TS 4.2 delivers new and enhanced capabilities in the area of scalability:
  - New Concurrency option for greater OTE exploitation
  - Threadsafe mirrors for function shipping over IPIC connections
  - Threadsafe CICS-DBCTL interface when connected to IMS V12
  - More threadsafe CICS API and SPI
  - New 64 bit infrastructure and exploitation
  - More VSAM LSR Pools

Notes

- We will start with a one page summary of scalability items in previous releases before moving on to cover the CICS TS 4.2 enhancements which centre on Open Transaction Environment (OTE) enhancements but also cover 64 bit exploitation and an enhancement to File Control to allow use of more VSAM LSR pools.

Scalability items in previous releases

- CICS TS 3.2
  - CICS-MQ exploitation of OTE
  - Threadsafe File Control (Post GA)
  - 64 bit container support
  - Extended ESDS support
  - Shared Data Tables > 2GB using multiple data spaces
  - XCF group limit changes

- CICS TS 4.1
  - Exploitation of z/OS XMLSS parser
  - IPv6 support
  - 64 bit positioning – extended MVS linkage support
  - Java positioning – JVMSERVER for dynamic scripting support

Notes

- Previous releases of CICS TS had differently named themes to group together enhancements but if we revisit CICS TS 3.2 and CICS TS 4.1 there are indeed a number of enhancements that can be reclassified under a scalability theme. This demonstrates an ongoing roadmap of enhancing CICS to allow to run more work, more quickly, in a region.

CICS TS 4.2
Threadsafe enhancements for multi-processor exploitation

Notes

- This page left intentionally blank.

OTE settings in CICS TS 4.1 & below

- Up till now you can define as program as
  - **concurrency(quaisirent)** meaning it must run on QR TCB
  - **concurrency(threadsafe)** meaning it can run on an open TCB or QR whichever is inuse at the time, it doesn't care which one is used
- There is no concurrency option to tell CICS that the application must run on an open tcb from the very start
- Instead, the only way you could do this was to define it with attribute **api(openapi)**
  - This has disadvantages (see next slide)

Notes

- A CONCURRENCY(QUASIRENT) program always runs on the QR TCB.
- A CONCURRENCY(THREADSAFE) program is a program that has been coded to threadsafe standards, contains threadsafe logic. It is capable of running on either the QR TCB or an open TCB. It starts off running on QR TCB. If processing such as a DB2 request causes a switch to an open TCB, then on return to the program the program continues on the open TCB.
- Prior to CICS TS 4.2, there is no concurrency option that specifies that the program should run on an open TCB from the start. Instead this can be achieved by defining it as API(OPENAPI) but this has disadvantages which are explained on the next slide.

OTE settings in CICS TS 4.1 & below

- Defining a program with attribute **api(openapi)**
  - Tells CICS the application will use non CICS apis
  - Has the side effect of running the application on an open TCB
  - Has drawback of forcing CICS to match execution key and TCB key so that non CICS apis will work. (This is not required for CICS apis)
  - Means CICS has to use L9 TCBs for user key programs
    - No good for DB2, MQ or sockets applications
      - Accessing a resource manager always uses an L8 TCB

Notes

- Defining a program with attribute API(OPENAPI) has the side effect of causing the program to run from the start on an open TCB. However, primarily, it is telling CICS that the program will issue non CICS supported API commands, ie something other than EXEC CICS, EXEC SQL, MQ commands etc, for example an MVS command.
- Unlike CICS commands, an MVS command for example will only work correctly when the key of the TCB matches the execution key. So when running EXECKEY(USER) which is key 9, an OPENAPI program is given a key 9 TCB called an L9 TCB.
- Contrast this with a program that only issues CICS supported commands. In this case CICS does not reference the key of the TCB, so a program can be running in userkey or CICSkey and can run under a key 8 TCB such as QR or an L8 TCB.
- Users who define programs as OPENAPI get the advantage of starting on an open TCB but suffer the disadvantage that an L9 will be used (assuming storage protection is active). This is a disadvantage because DB2 requires an L8 TCB and so we switch from L9 to L8 and back again for every DB2 request. This is why OPENAPI should not be used for DB2 programs, or MQ programs.

OTE settings in CICS TS 4.2

- In CICS TS 4.2 we have separated out whether an application must run on an open TCB from what type of APIs it uses
- CONCURRENCY(REQUIRED)
  - Application must be coded to threadsafe standards
  - States that the application MUST run on an open TCB
  - Application starts on an open TCB
  - If a switch to QR is made for a CICS command, a switch back to the open TCB is made when returning to the application
- Existing API keyword defines what APIS are used
  - This defines what type of TCB is used

Notes

- CICS TS 4.2 provides a new CONCURRENCY(REQUIRED) setting which specifies that the program requires to run on an open TCB. It will run an open TCB from the start, and if CICS has to switch to QR TCB to process a non-threadsafe CICS command, it will return to the open TCB when it returns to the application program. Now the user can define that the program must start on an open TCB, independently of defining what APIs it uses.
- The API parameter determines what type of open TCB is used.

OTE settings in CICS TS 4.2

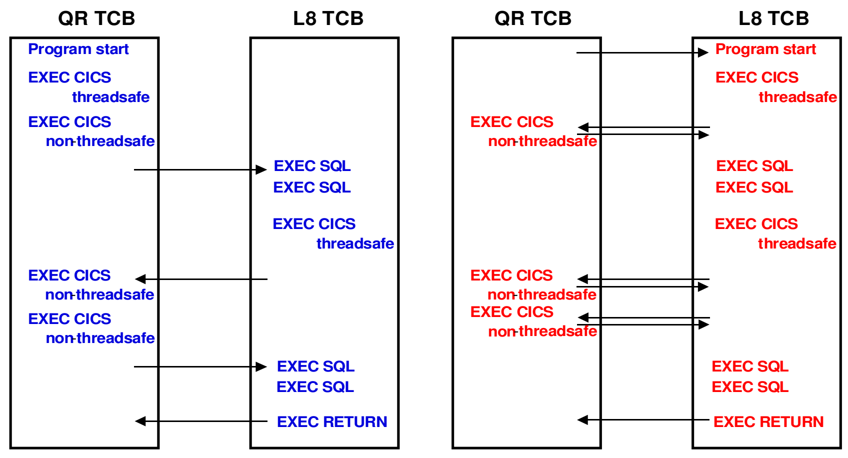
- CONCURRENCY(REQUIRED) & API(CICSAPI)
  - the application will run on an open TCB from the start.
  - It only uses CICS supported apis (including DB2, IMS & MQ)
    - CICS will always use an L8 TCB in this instance irrespective of the execution key, as CICS apis do not rely on the key of the TCB.
  - This is great for applications that are going to resource managers like DB2 & MQ as the same L8 is used

- CONCURRENCY(REQUIRED) & API(OPENAPI)
  - the application will run on an open TCB from the start.
  - As it will use non CICS apis it will run on an L8 or an L9 TCB depending on the execution key. This is the same as with CICS TS 4.1 today
  - Only use OPENAPI when non CICS supported apis are to be used

Notes

- For CONCURRENCY(REQUIRED) programs the type of open TCB used depends on what APIs the program is to use:
  - If the program uses only CICS supported APIs (including access to external resource managers such as DB2 IMS and WebSphere MQ) then it should be defined with program attribute API(CICSAPI). In this case CICS always uses an L8 open TCB, irrespective of the execution key of the program, because CICS commands do not rely on the key of the TCB.
  - If the program is to use other non-CICS APIs then it must be defined with program attribute API(OPENAPI). In this case CICS uses an L9 TCB or an L8 TCB depending on the execution key of the program. This is to allow the non-CICS APIs to operate correctly. This OPENAPI behaviour is the same as previous releases.

CONCURRENCY(THREADSAFE) vs CONCURRENCENCY(REQUIRED)



The program for this transaction is defined
CONCURRENCY(THREADSAFE)
API (CICSAPI),

EXECKEY(USER) or EXECKEY(CICS)

The program for this transaction is defined
CONCURRENCY(REQUIRED)
API (CICSAPI),

EXECKEY(USER) or EXECKEY(CICS)

Notes

- Existing threadsafe CICS-DB2 applications, which have taken advantage of the performance gains of being able to run on the same TCB as the DB2 call by being defined as THREADSAFE CICSAPI, can be further enhanced by defining them as REQUIRED CICSAPI. This definition means that the programs can run on an L8 open TCB (irrespective of their execution key as CICS can for example run the program in key 9 on an key 8 TCB) from the start without waiting for the first DB2 call to move them on to the open TCB. Achieving additional benefit depends on how many, if any, non-threadsafe CICS commands the application executes

Threadsafe Mirror

- DFHMIRS is now Threadsafe
  - Supplied definition now specifies CONCURRENCY(THREADSAFE)

- IPIC transformers are now threadsafe. Non IPIC code remains non threadsafe

- Only requests function shipped over IPIC will run on an Open TCB
  - File Control & Temporary Storage
  - Distributed Program Link
    - If the target program is defined as threadsafe and the mirror already on an open TCB

- Review your DFHSIT specification for FCQRONLY if using IPIC
  - Specify FCQRONLY=NO as there is no longer any need to turn off threadsafety in the FOR

Notes

- The CICS-supplied mirror program DFHMIRS, which is used by all mirror transactions, is now defined as threadsafe. In addition the IPIC transformers have been made threadsafe. For IPIC connections only, CICS runs the mirror program on an L8 open TCB whenever possible. For threadsafe applications that function ship commands to other CICS regions using IPIC, the resulting reduction in TCB switching improves the performance of the application compared to other intercommunication methods. To gain the performance improvement, you must specify the system initialization parameter FCQRONLY=NO in the file-owning region.

- File control requests that are function shipped using IPIC connectivity provide threadsafe file control with significant potential throughput improvements over LU6.2 in CICS regions with multiple processors available.

- Temporary storage requests that are function shipped using IPIC connectivity are threadsafe and no longer need to switch to QR before being function shipped.

Threadsafe Function shipping over IPIC

- Remote File Control and TS requests over IPIC are now threadsafe
  - No switch to QR in the AOR
  - Transformers can run on open TCB in the AOR and ship the request

- File Control and TS requests in the remote region can run on the open TCB
  - Mirror switches to open TCB for 1st FC or TS request
  - TS request will run on open TCB
  - FC request will run on open TCB provided:
    - FCQRONLY=NO is set
    - Its local VSAM LSR file or RLS (ie existing FC threadsafe support)

- MIRRORLIFE on the IPCONN defn controls lifetime of mirror
  - UOW or TASK allows mirror to continue

Notes

- For remote File Control or TS requests shipped over IPIC connections, CICS will nolonger force a switch to QR TCB if it is running currently on an open TCB. The requests will be shipped running on the open TCB.
- In the FOR or QOR, the mirror decides when to switch to an open TCB. It does so for the first File Control or TS request received over an IPIC connection. The idea is for long running mirrors to keep the mirror running on an open TCB.
- A new option MIRRORLIFE has been added to the IPCONN attributes for function-shipped file control, transient data, and temporary storage requests using an IPIC connection. MIRRORLIFE improves efficiency and provides performance benefits by specifying the lifetime of mirror tasks and the amount of time a session is held.

Threadsafe CICS-DBCTL

- CICS-DBCTL interface will use OTE when connected to IMS 12*
  - At connect time CICS & IMS determine if each other can support OTE
  - With IMS 10 & 11
    - CICS-DBCTL TRUE enabled as QUASIRENT
    - Toleration IMS V10 apar PM31730 or IMS V11 apar PM31729
  - With IMS 12*
    - CICS-DBCTL TRUE enabled as OPENAPI
    - Apply CICS TS 4.2 fix PM42781
    - IMS V12 Exploitation apar PM31420 is required
- * At General Availability of CICS TS V4.2, IMS 12 is available through a Quality Partnership Program (QPP).
  - For more information, visit http://www.ibm.com/software/data/ims/

Notes

- CICS provides a CICS IMS Database control (CICS-DBCTL) interface for IMS to satisfy DL/I requests that are issued by applications running in a CICS region. In CICS TS 4.2 the CICS-DBCTL interface has been defined as threadsafe and CICS can run the CICS-DBCTL task-related user exit (TRUE) on an L8 open task control block (open TCB).
- The open transaction environment (OTE) is supported from IMS version 12 with PTFs for APAR PM31420 applied. IMS indicates to CICS during the connection process that the OTE is supported and consequently CICS defines the CICS-DBCTL TRUE as an open API TRUE.
- An open API TRUE is run on an L8 open TCB, which is dedicated for use by the calling CICS task. Running an application on an open TCB improves throughput and performance by reducing the use of the QR TCB. Threadsafe CICS applications that run on an L8 open TCB and use threadsafe CICS-DBCTL commands now avoid up to four TCB switches for each call to IMS. For more information about CICS IMS applications and the OTE, see Enabling CICS IMS applications to use the open transaction environment (OTE) through threadsafe programming.
- If your IMS version does not support the OTE, CICS runs the CICS-DBCTL TRUE on the QR TCB.

Threadsafe CICS-DBCTL

- TCB switching is controlled by the IMS Database Resource Adapter (DRA)
  - IMS 12 DRA will not switch TCBs and run the IMS request on the calling L8 TCB
  - < IMS 12 DRA will continue to be called on QR TCB and will switch to a DRA thread TCB
- CICS code implementing CICS-DBCTL interface made threadsafe
- CALLDLI and EXEC DLI api now threadsafe when run with IMS 12

Notes

- The IMS DRA is the code that prior to CICS TS 4.2 will switch from QR TCB to an IMS thread TCB and back again for each IMS request. With CICS TS 4.2 and IMS V12 the DRA will not switch TCBs but will run the request on the calling TCB which will be an L8 TCB. Hence the CALL DLI and EXEC DLI commands are threadsafe when running with IMS V12 and will run on the L8 TCB.
- The CICS side of the CICS-DBCTL interface has been made threadsafe so that the processing can run on an L8 TCB.

Threadsafe CICS-DBCTL

- For non threadsafe CICS applications there is no change to the amount of TCB switching
  - Switch to L8 to invoke DFHDBAT and IMS
  - Switch back to QR when returning to the application
- For threadsafe CICS applications save TCB switches
  - Remain on L8 after return from IMS request
  - Expect same benefits as seen for threadsafe CICS-DB2 applications
    - Reduced cpu
    - More throughput

Notes

- For a non threadsafe application there is no reduction in the amount of switching. Instead of switching from QR TCB to an IMS thread TCB abd back again for each IMS request, it switches from QR to L8 and back again.
- For a threadsafe application, If it is running on QR TCB it will switch to L8 and then stay on L8 when returning to the application.
- For a threadsafe application that is already running on an L8 TCB, or for a CONCURRENCY(REQUIRED) application running on an L8 TCB then no tcb switching occurs for the IMS request.

Threadsafe CICS-DBCTL

- CICS-IMS applications when using IMS 12 will use L8 TCBs
  - Adjust MAXOPENTCBs as necessary
- DRA startup table DFSPZPxx still controls how many IMS threads can be used
  - MINTHRDS - the number of threads that remain 'signed on'
    - DRA keeps IMS threads but dissociated from the TCB
  - MAXTHRDs will still limit the number of IMS threads from this CICS region
    - Even though an L8 TCB may be available, IMS limits how many can be used for threads
      - If exceed MAXTHRDS will queue in the IMS DRA as it does today (pre OTE)

Notes

- MINTHRD=xxx
  - This parameter specifies the number of threads for this CICS system that, once initialized, remain created while the DRA is active. These threads remain allocated until this CICS system is disconnected from DBCTL, except if a thread is stopped by a /STOP command or by a thread failure. Additional threads are created, up to the number specified in MAXTHRD, or the number specified in MAXREGN, or the maximum of 999, whichever of these values is the lowest. These additional threads (not the MINTHRDs) are released when there is not enough system activity to require them. The maximum value you can specify for MINTHRD is 999, and the default is 1. For information about specifying values for MINTHRD, see Specifying numbers of threads. See also MAXREGN in IMS system generation macros used by DBCTL.
- MAXTHRD=xxx
  - This parameter specifies the maximum number of transactions for which this CICS system can have PSBs scheduled in DBCTL. Any schedule requests that are over this limit are queued in the DRA. You can balance the load sent to a single DBCTL from multiple CICS systems by specifying appropriate values for MAXTHRD in each CICS.
    The maximum value you can specify for MAXTHRD is 999 (but it should not exceed the value specified for MAXREGN) and the default is 1, or the value you specified in MINTHRD. For information about specifying values for MAXTHRD, see Specifying numbers of threads. See also MAXREGN in IMS system generation macros used by DBCTL.

Threadsafe syncpoint

- Commands now made threadsafe*
  - EXEC CICS SYNCPOINT
  - EXEC CICS SYNCPOINT ROLLBACK
  - EXEC CICS RESYNC
    * some switching may still occur, but heavily reduced

- EXEC API nolonger switches to QR for syncpoint requests
  - Recovery Manager (RM) domain now makes the decision if a switch is required
  - All RM domain clients register at startup and tell RM if they are threadsafe
  - RM domain keeps it on the open TCB wherever possible

- End of task syncpoint can run on an open TCB before CICS switches to QR to terminate the task

Notes

- The Recovery Manager processes this command on an open TCB wherever possible to minimize TCB switching. Syncpoint processing can take place on an open TCB for all resource types declared as threadsafe that were accessed in the unit of work. If resource types not declared as threadsafe were accessed in the unit of work, the Recovery Manager switches to the QR TCB for those resource types. A CICS resource type declares itself to the Recovery Manager as threadsafe if the EXEC CICS commands relating to the resource type are threadsafe

- Prior to CICS TS 4.2, CICS would switch to QR prior to end of task syncpoint. In CICS TS 4.2 it remains on an open TCB, if it is running on one, until end of task syncpoint has been called. Afterwards it switches to QR for the task detach logic.

Threadsafe syncpoint

- Non-threadsafe Recovery Manager clients will force a switch to the QR TCB
  - Examples:
    - Local Terminal, MRO, Transient Data, etc
- Recovery Manager clients may themselves switch to the QR TCB
  - Examples:
    - File control for BDAM files
    - RMI for a quasi-reentrant TRUE
- Example, threadsafe application running on an open TCB that has updated DB2 and MQ issues a syncpoint:
  - In CICS TS 4.1: 9 TCB switches occur
  - In CICS TS 4.2: 0 or 1 switch occurs

Notes

- Terminal driven transactions involve CICS terminal control logic which is not threadsafe, so a switch to QR will occur during phase 2 syncpoint processing. This is an example of Recovery manager clients that will cause a switch to QR.
- Other RM cleints will themselves switch to QR if necessary For example, File Control is threadsafe when processing local vsam files, but it will switch to QR TCB for BDAM files.
- Prior to CICS TS 4.2 a threadsafe application running on an L8 TCB that had updated DB2 and MQ and then issued a syncpoint would suffer 9 TCB switches.
  - A switch to QR would be made at the start of syncpoint.
  - Switches to L8 and back to QR would occur when calling DB2 for PREPARE
  - Switches to L8 and back to QR would occur when calling MQ for PREPARE
  - Switches to L8 and back to QR would occur when calling DB2 for COMMIT
  - Switches to L8 and back to QR would occur when calling MQ for COMMIT
- In CICS TS 4.2, if it was a terminal driven transaction one TCB switch to QR would occur. For a non terminal driven transaction (and assuming no other non threadsafe resources had been touched) then no TCB switches will occur.

Other APIs made threadsafe

- QUERY SECURITY
- SIGNON, SIGNOFF
- VERIFY PASSWORD, VERIFY PHRASE
- CHANGE PASSWORD, CHANGE PHRASE

- EXTRACT TCPIP, EXTRACT CERTIFICATE

- All Call and Exec Level Named Counter Server commands
- Built in functions for DIGEST and DEEDIT

Notes

- The listed API commands have been made threadsafe in CICS TS 4.2 and can run on an open TCB.

Threadsafe SPI commands

- New SPI commands that are threadsafe:
  - INQUIRE CAPDATAPRED, INQUIRE CAPINFOSRCE, INQUIRE CAPOPTPRED
  - INQUIRE EPADAPTER, SET EPADAPTER
  - INQUIRE OSGIBUNDLE, INQUIRE OSGISERVICE
  - INQUIRE TEMPSTORAGE, SET TEMPSTORAGE
- Existing SPI commands made threadsafe:
  - INQUIRE CLASSCACHE
  - INQUIRE JVM
  - INQUIRE JVMPOOL
  - INQUIRE JVMPROFILE
  - PERFORM CLASSCACHE
  - PERFORM JVM POOL
  - SET CLASSCACHE
  - SET JVMPOOL

Notes

- The listed SPI commands have been made threadsafe in CICS TS 4.2 and can run on an open TCB.

CICS TS 4.2

64 bit infrastructure & exploitation

Notes

- This page left intentionally blank.

CICS and 64-Bit Support : Overview

- Enabled the CICS domain architecture to run in, and exploit 64-bit addressing mode

- Allowed CICS domains to use stack storage, domain anchor storage, and all associated domain control blocks in virtual storage above the bar

- Provided below the bar VSCR …
  - Single System Scaling …
    - More concurrent tasks, larger applications, …
  - Increasing pressure on storage usage above the 16MB line!
    - Both in z/OS storage and in EDSA storage!

Notes

- The aim of the CICS 64-bit Re-Architecture is to provide a CICS domain architecture environment that exploits the underlying z/Architecture for 64-bit addressing and to provide the infrastructure for the future so that CICS applications will be able to utilize and exploit 64-bit addressing mode. This will enable CICS to remove some of the previous limitations that affect scalability and availability by delivering large address spaces with the exploitation of the 64-bit addressing provided by the z/Architecture. With z/OS 64-bit virtual storage, CICS can make use of this large 64-bit virtual storage to increase capacity by supporting a larger number of concurrent users and concurrent transactions as well as keeping up with the virtual storage demands of increased workload of existing applications and the larger memory requirements of new applications and new technologies.

CICS and 64-Bit Support : Overview

- CICS Transaction Server for z/OS V4.2 contains significant changes to the CICS domain architecture that exploit the underlying z/Architecture for 64-bit addressing and provides the infrastructure for CICS domains to utilize and AMODE 64.

- These changes affect a number of CICS internal interfaces and related control blocks, including (but not limited to)
  - CICS Kernel Anchor and Domain Table separated, …
  - Kernel Stack, Stack Segments, Larger Save Areas, …
  - Kernel Task Entry (TAS), …
  - Common System Area (CSA) – significantly changed, …
  - Task Control Area (TCA) – 15% smaller, …
  - Control block changes for the following domains
    - Trace, Message, Loader, Temporary storage, Monitoring...

Notes

- In CICS TS 4.2 we have enabled the CICS domain architecture to run in, and exploit, 64-bit addressing mode, allowing CICS domains to use stack storage, domain anchor storage, and all associated domain control blocks in virtual storage above the bar.

CICS and 64-Bit Support : Overview

- The following domains run AMODE(64)
  - Trace Domain
  - Message Domain
  - Temporary Storage Domain
  - Kernel Domain
  - Monitoring Domain
  - Storage Manager Domain
  - Lock Manager Domain

- MEMLIMIT
  - New minimum requirement of 4G
    - CICS will not start with less - message DFHSM0602

Notes

- The listed domains nor run AMODE(64).
- The z/OS MEMLIMIT parameter limits the amount of 64-bit (above-the-bar) storage that the CICS address space can use. This storage includes the CICS dynamic storage areas above the bar (collectively called the GDSA) and MVS storage in the CICS region outside the GDSA.
- A CICS region requires at least 4 GB of 64-bit storage. You cannot start a CICS region with a MEMLIMIT value that is lower than 4 GB. If you attempt to do so, message DFHSM0602 is issued, a system dump with the dump code KERNDUMP is produced, and CICS terminates. Note: CICS does not try to obtain the MEMLIMIT amount of storage when initializing. 64 bit storage is obtained as required.

Storage Manager Domain

- Improved 64-Bit Storage Manager Statistics …
  - Additional 64-bit Storage Usage Metrics from the z/OS RAX
    - RAX ➠ z/OS RSM Address Space Block Extension
  - Number of Memory Objects, Number of Shared Memory Objects,
  - Allocated private storage objects, high-water-mark, …
  - Allocated shared storage objects, high-water-mark, …
  - Current and Peak GDSA allocated, …
- SOS infrastructure for above the bar storage implemented
  - SMNT Storage_Notify infrastructure implemented
    - RS Domain now Notified when SOS occurs – rather than noticing!
  - For future exploitation – No current CICS 64-bit exploiter can do much about it!
- Metrics useful to measure the impact of 64 bit JVMs
  - Storage for 64 bit JVMs is outside of the CICS DSAs

Notes

- The CICS storage manager statistics now provide additional information about 64-bit storage. The storage manager global statistics, mapped by the DFHSMSDS DSECT and the storage manager dynamic storage areas statistics, mapped by the DFHSMSDS DSECT have been enhanced.
- The reports produced by the DFHSTUP and DFH0STAT statistics programs show the new statistics.

CICS and 64-Bit Support : Effects on 31 bit storage

- EDSALIM
  - Changes in CICS over the years mean that minimum and default EDSALIM sizes are no longer adequate.
  - Current values
    - Minimum size is 10M
    - Default size is 34M
  - New values
    - Minimum size is 48M
    - Default size is 48M
    - Maximum size is unchanged at 2047M
  - Changes reflected in supplied SITs and IVPs

Notes

- The minimum and default EDSALIM values have changed to 48 MB to ensure that there is sufficient storage for CICS initialization.
- The EDSALIM system initialization parameter specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside in 31-bit (above-the-line) storage; that is, above 16 MB but below 2 GB.

CICS and 64-Bit Support : Exploiters

- CICS Java
  - Move to 64-bit JVM (Java 6) for pooled JVM and JVMServer
  - 31-bit Java not supported

- CICS Trace
  - Internal Trace Table above the bar
    - Internal trace table in 64-bit storage only if Transaction Isolation inactive (TRANISO=NO) *or* APAR OA34311 applied on z/OS 1.12
  - Transaction Dump Trace Table in 64-bit storage
  - Many trace control blocks above the bar
  - 64-bit GTF tracing (OA32611 required for z/OS 1.11 & 1.12)

- Message tables are above the bar (subject to TRANISO restriction)

Notes

- All JVMs now run in AMODE(64) instead of AMODE(31), increasing the capacity for running more JVMs in a CICS region. JVM servers and pooled JVMs use 64-bit storage, significantly reducing the storage constraints in a CICS region for running Java applications. You can therefore reduce the number of CICS regions that run Java to simplify system management and reduce infrastructure costs. You can also use System z Application Assist Processors (zAAPs) to run eligible Java workloads. CICS uses the IBM 64-bit SDK for z/OS, Java Technology Edition, Version 6.0.1. You must download this version of the SDK to run Java applications in CICS. You can continue to build Java programs using a different version of the SDK. If you have any programs that use the Java Native Interface (JNI), including other products, you must ensure that these programs can run in a 64-bit environment.

- CICS can obtain 64-bit (above-the-bar) storage, rather than 31-bit (above-the-line) storage for the internal trace table, depending on the version of the z/OS operating system, and whether the CICS region operates with transaction isolation.

CICS and 64-Bit Support : Exploiters

- CICS DUMP
  - 64-bit storage SDUMP support
    - Requires z/OS APAR OA32271 (z/OS 1.11 and 1.12)
  - Use is now made of DUMPRIORITY for key control blocks in 64-bit storage that should have priority when taking an SDUMP
    - Improves chances of storage being in the SDUMP in case the dump data sets are not big enough!
    - Recommend always use SMS managed dump data sets

- EXITS
  - XPI is based on domain architected interfaces
    - Linkage conventions have changed
  - All Global user exits using the XPI MUST be reassembled

Notes

- CICS uses the DUMPRIORITY keyword to ensure that key control blocks, like the trace table, are captured in a dump. You are recommended to use SMS managed dump data sets so that the dump datasets are big enough.

- The Exit Programming Interface (XPI) used in Global user exits is based on the CICS domain architecture. Because significant changes have been made to the domain architectuire and the linkage conventions used, all exits that contain XPI calls must be reassembled.

CICS and 64-Bit Support : Temporary Storage

- Many TS control blocks above the bar
- TS Main above the bar (subject to TRANISO restriction)
- Limiting Main Temporary Storage use
  - A new SIT parameter TSMAINLIMIT={64M|amount}
  - Maximum is 32G, but must not be greater than 25% of MEMLIMIT else CICS will not start - message DFHTS1608
  - Available on INQUIRE and SET TEMPSTORAGE commands
    - SET TEMPSTORAGE TSMAINLIMIT cannot decrease by too much or increase to >25% of MEMLIMIT
      - INVREQ and message DFHTS1606 or DFHTS1607
  - TSMAININUSE option on INQUIRE TEMPSTORAGE provides current amount of storage occupied

Notes

- Main temporary storage queues can now use 64-bit (above-the-bar) storage. CICS provides new facilities so that you can check the storage use of main temporary storage queues and limit that storage use.
- Main temporary storage is in 64-bit storage rather than 31-bit (above-the-line) storage, depending on the version of the z/OS operating system and whether the CICS region operates with transaction isolation.
- If your CICS applications use large amounts of main temporary storage, the move to 64-bit storage can increase the available storage elsewhere in your CICS regions.
- Auxiliary temporary storage queues and shared temporary storage queues continue to use 31-bit storage.
- If your CICS applications currently specify the location of temporary storage, you can review this. If an application specifies that auxiliary temporary storage is used, and you do not require recoverable temporary storage, you can change it to specify main temporary storage. The advantages of this action are that space becomes available in 31-bit storage, and input/output activity to write data to disk is reduced.
- You can control how much storage the CICS region makes available to main temporary storage queues by using the TSMAINLIMIT system initialization parameter. The default is 64 MB. This limit on storage use does not include auxiliary temporary storage queues and shared temporary storage queues.

Temporary Storage : Automatic TS cleanup

- EXPIRYINT can be specified on a TSMODEL definition
  - Applies to
    - TS Main and non-recoverable Aux
  - Does NOT apply to
    - Remote queues – use model in QOR instead
    - Recoverable AUX
    - Shared TS queues
    - CICS internal queue
    - Queues that don't match a TS model
  - Default is zero (no expiry), interval can be specified in hours
    - 0 ¦ 1-15000
  - Available via CEDA, CSDUP, CPSM BAS and Explorer
  - SPI to inquire and set and via CPSM and Explorer

Notes

- You can specify that CICS deletes temporary storage queues automatically when they are no longer required.
- Automatic deletion of eligible temporary storage queues reduces the unnecessary use of virtual storage. To use this feature, you set suitable expiry intervals in the temporary storage models (TSMODEL resource definitions). The expiry interval is available for main temporary storage queues and nonrecoverable auxiliary temporary storage queues that match TSMODEL resource definitions in the local CICS region.

Temporary Storage : Automatic TS cleanup

- CICS system task attached every 30 minutes
  - Uses algorithm to minimise how often full scan of queues is performed
  - Quick scan of models to pick up additions/deletions
  - If no TSMODELs with an expiry interval task terminates immediately
  - Not an exact match on expiry interval
    - Queues will be deleted by interval+10% subject to 30 minute minimum
  - When performs a scan, it issues message DFHTS1605.
    - shows the number of temporary storage queues that were scanned and the number that were deleted.
  - If the cleanup task ends abnormally, it issues message DFHTS0001 and does not run again until CICS is restarted.

Notes

- EXPIRYINT specifies the expiry interval, in hours, for a temporary storage queue that matches this model. The interval count begins after each use of the temporary storage queue. If the queue is not used again before the expiry interval is reached, the queue becomes eligible for CICS to delete it automatically.
- When the CICS cleanup task performs a scan, it issues message DFHTS1605. This message shows the number of temporary storage queues that were scanned and the number that were deleted. If the cleanup task ends abnormally, it issues message DFHTS0001, and does not run again until CICS is restarted.

Temporary Storage : Automatic TS cleanup...

- If you change the expiry interval in a TSMODEL
  - existing temporary storage queues that match the model are not affected – use expiry interval that applied when they were created

- Existing TSAGE TST parameter unaffected
  - If your CICS region still uses a temporary storage table (TST), which can be used in combination with TSMODEL resource definitions
    - the TST might include a TSAGE parameter. TSAGE specifies an aging limit in days, up to 512 days, for temporary storage queues.
    - If the TST includes a nonzero TSAGE and there is an emergency restart of CICS, CICS deletes temporary storage queues that were not referenced during the specified interval
    - The TSAGE parameter does not cause automatic deletion of queues at any other time.

Notes

- If you change the expiry interval in a TSMODEL resource definition, existing temporary storage queues that match the model are not affected. Those queues continue to use the expiry interval that applied when they were created. If all the TSMODEL resource definitions with a nonzero expiry interval are deleted from a CICS region, CICS stops scanning for expired temporary storage queues.

- The existing behaviour of the TSAGE parameter specified on a macro TST is unchanged.

CICS TS 4.2
More VSAM lsrpools

Notes

- This page left intentionally blank.

More VSAM LSR Pools

- CICS historically has allowed use of 8 LSR pools
  - In the days of local DLI, DLI used the other 8
  - When local DLI was removed, CICS File Control was not changed
- Since z/OS 1.4, DFSMS have allowed up to 256 lsrpools per address space.
- In CICS TS 4.2 we will allow use of 255 LSRPOOLS (1-255)
  - We will not use pool 0
  - Today LSRPOOL number is held in one byte in the FCTE, with 0 meaning no LSR (ie NSR)
- Potential performance optimization where greater subdivision of files across LSRPOOLs is required
  - e.g. Place highly-used files in their own LSRPOOL

Notes

- You can now define up to 255 LSR pools using the LSRPOOLNUM attribute in the FILE and LSRPOOL resource definitions.
- The existing attribute LSRPOOLID in the FILE and LSRPOOL resource definitions has been replaced with LSRPOOLNUM to increase the maximum number of LSR pools available in a CICS region from 8 to 255.
- The LSRPOOL resource defines the size and characteristics of the local shared resource (LSR) pool. Increasing the number of LSR pools available in a CICS region can improve transaction response times for CICS workloads that use VSAM files.
  - For example an application that processes an entire data set every time the transaction is used can cause excessive I/O operations. Placing the data set into its own LSR pool can help eliminate I/Os and prevent buffer stealing in a shared LSR pool.
  - You can also slow down transactions that are causing buffer steals in a shared VSAM LSR pool by placing the VSAM data sets in a detuned LSR pool to limit the number of data component buffers. In many cases a dedicated LSR pool performs better than a shared LSR pool. Dedicated LSR pools are also an excellent replacement for CICS data tables, when the capacity of the CICS data tables has been exceeded.

More VSAM LSR Pools

- CEDA DEFINE FILE & DEFINE LSRPOOL
  - Today CEDA uses one character field for LSRPOOLID
  - New LSRPOOLNUM keyword introduced allowing values NONE¦ 1-255. Default is 1
  - LSRPOOLID now only set via compatibility mode for back level systems with range 1-8
  - For existing definitions, the value of LSRPOOLID is transferred to LSRPOOLNUM

- DFHCSDUP
  - Allows specification of LSRPOOLNUM with range 1-255
  - Allows setting of LSRPOOLID with range 1-8 in compatibility mode
  - Existing jobs using LSRPOOLID will run – value transferred to LSRPOOLNUM

Notes

- You can use the new LSRPOOLNUM attribute on the FILE and LSRPOOL resource definitions to specify LSR pool numbers in the range 1 – 255. The existing LSRPOOLID value which takes values in the range 1 – 8 is maintained for compatibility.

More VSAM LSR Pools

- EXEC CICS INQUIRE/SET FILE
  - LSRPOOLNUM defined as an alias of LSRPOOLID. Existing programs using LSRPOOLID will still work
- EXEC CICS CREATE FILE & CREATE LSRPOOL
  - Supports use of LSRPOOLNUM. Existing use of LSRPOOLID will still work
- SIT
  - Existing parameter CSDLSRNO changed to support a range of 1-255
- Explorer
  - Supports LSRPOOLNUM for operations and admin
- CSPM
  - Supports LSRPOOLNUM for operations and ADMIN
- STATS
  - Report on up to 255 pools

Notes

- All Externals that today use LSRPOOLID in the rnage 1-8 have been changed to support LSRPOOLNUM in the rnage 1-255.

Further Reading

- White paper: IBM CICS Scalability: New features in V4.2
  - ftp://public.dhe.ibm.com/software/htp/cics/pdf/Scalability_paper_final.pdf

Copyright and Trademarks