Analyzing CICS TS SOS Problems in z/VSE

Mike Poil CICS L3 Service

http://www.ibm.com/zVSE

http://twitter.com/IBMzVSE

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

References

- CICS TS for VSE/ESA Performance Guide SC33-1667
- CICS TS for VSE/ESA Problem Determination Guide SC33-1663
- CICS TS for VSE/ESA Trace Entries SC33-6108.
- CICS TS for VSE/ESA Enhancements Guide GC34-5763
- CICS TS for VSE/ESA Operations and Utilities Guide SC33-1654
- z/VSE LVC June 2013 How to handle or avoid CICS storage availability problems
- WAVV 2013 CICS Performance Workshop
- WAVV 2014 update for WAVV 2012 How to Monitor and Optimize CICS Storage
- WAVV 2014 Introduction to CICS TS Storage Manager by Gene Hudders

Abstract

This session begins with a short introduction to the design of CICS storage management. It explains what triggers an SOS condition, how it affects CICS processing and introduces a potential workaround. It identifies possible reasons for SOS occurring and actions to resolve them. It explains what is required to be in place to capture the correct diagnostic data for problem determination, and shows how to use DFHPD410 formatted dump output in conjunction with the appropriate CICS manuals to find the root cause and hence pursue the appropriate resolution.

Agenda

- CICS DSA overview.
- CICS subpools.
- CICS storage requests.
- What is Short-On-Storage?
- The SIT MXT parameter and SOS.
- Can I do anything to work around SOS?
- Could I have prevented SOS?
- What can cause SOS?
- A Storage Leak.
- How do I resolve SOS?
- What diagnostic material do I need?

Agenda

- What signs do I look for in a dump or statistics?
- Analyzing SOS problems.
- Problem #1.
- Problem #2.
- Problem #3.
- Q & A.

CICS DSA Overview

- Most of the CICS storage requirement is managed through its DSAs.
- There are 4 types of DSA, with 24-bit (Below 16MB) and "E" 31-bit (Above 16MB) versions.
- The CDSA and ECDSA are for CICS-key storage requirements.
  - CICS control blocks.
  - Non-reentrant CICS nucleus programs.
  - Non-reentrant EXECKEY(CICS) programs.
  - Other CICS-key storage.
  - CICS-key is the protection key of the Partition.
- The RDSA and ERDSA contain reentrant (SVA-eligible) programs when the SVA copies are not being used by CICS (SIT SVA=NO etc.).
  - CICS nucleus and programs defined in the CSD and by Program Autoinstall.
  - SIT RENTPGM=PROTECT is recommended, and uses storage key 0 protection for each phase's code rather than using the less secure CICS-key.
  - *Note: Phase DFHSIP31 needs to be loaded into the SVA and SIT SVA=YES must be used to protect it with key 0 storage.*

CICS DSA Overview

- The SDSA and ESDSA are for shared USER-key storage requirements.
  - EXEC CICS GETMAIN SHARED.
  - Non-reentrant EXECKEY(USER) programs.
  - Other data areas.
  - USER-key access, which is key 9 if SIT STGPROT=YES, and the protection key of the Partition if STGPROT=NO.
  - The use of STGPROT=YES is recommended.
- The UDSA and EUDSA are for non-shared USER-key storage requirements.
  - USER-key program task-related storage.

CICS DSA Overview

- SIT DSALIM (24-bit) and EDSALIM (31-bit) define the limits for DSA storage.
- The amounts are allocated from contiguous Partition GETVIS, and are mapped internally as a series of 256K and 1MB EXTENTs respectively that belong to the "available extent" pool.
- *Note: the two GETVIS storage area remain allocated until CICS terminates even if the whole amounts are not allocated for DSAs to use*.
- One or a series of contiguous extents are allocated to a DSA when it needs to expand.
- An extent normally remains allocated to a DSA even if it becomes empty, hence the amount of used (E)DSALIM will grow to a peak value based on concurrent demands.
- A DSA's size will contract if an empty extent is transferred to another DSA as part of SOS avoidance, but this will not affect peak (E)DSALIM usage.
- DSALIM and EDSALIM can be increased by CEMT I DSA, but only if there is contiguous free 24-bit or 31-bit (not "ANY") GETVIS storage respectively.
- *Note: Over-committing GETVIS is a potentially fatal condition just like SOS.*
- *Note: You can't start a new CEMT task when CICS is at SOS.*
- Reducing then increasing (E)DSALIM by CEMT I DSA may return empty extents to the "available extent" pool, but be careful if you try this at home!

CICS DSA Overview

- A DSA contains SUBPOOLs, each of which has a specific purpose.
- CICS System Subpools are documented in Appendix C of the CICS Performance Guide.
- Every task has 4 TASK subpools allocated for its (E)CDSA and (E)UDSA storage, and the subpool name is based on the DSA (M, C, B or U) and the 7-digit task number.
- A subpool is a series of 4K pages, but EUDSA (Unnnnnnn) task subpools use 64K pages.
- A storage request is mapped as an ELEMENT into one or more contiguous pages using First Fit logic.
- However, selected CICS system subpools are mapped as Quick-Cells that contain fixed-length elements of a CICS-determined size - Quick-Cells require less cpu to manage them.
- The CICS design will produce some fragmentation, leaving "holes" that cannot be reused, but this is normal for any product that provides storage management.
- DSA usage is also a function of the way that programs are written, and on the configuration options being used for CICS and its resource definitions.

CICS Subpools

- Examples of 24-bit subpools:
  - LD subpool sizes are based on the number and size of programs that are currently *loaded* in storage; a large amount of storage would normally be a function of what you asked CICS to support.
    - LDNRS - 24-bit CICS key programs that are not SVA-eligible.
    - LDNUC - 24-bit CICS nucleus programs that are not SVA-eligible.
    - LDPGM - 24-bit USER key programs that are not SVA-eligible.
    - LDRES - 24-bit USER key RESIDENT programs that are not SVA-eligible.
    - LDNRSRO - 24-bit CICS key programs that are SVA-eligible.
    - LDNUCRO - 24-bit CICS nucleus programs that are SVA-eligible.
    - LDPGMRO - 24-bit USER key programs that are SVA-eligible.
    - LDRESRO - 24-bit USER key RESIDENT programs that are SVA-eligible.
  - SMSHRU24 - 24-bit GETMAIN SHARED.

CICS Subpools

- Examples of 31-bit subpools:
  - LDExxxxx Loader Domain subpools.
  - ARI0OLRM - used for DB2/VSE.
  - DFHTDG31 - Transient Data general storage and control blocks based on SIT TD=.
  - DFHTDIOB - Transient Data buffers based on SIT TD=.
  - JCDYNLOG - CICS Dynamic Log for backout of recoverable resources; the lifetime of this type of storage for a task is syncpoint-related.
  - SMTP - Terminal I/O areas based on the number of terminals and activity.
  - SMSHRU31 - 31-bit GETMAIN SHARED.
  - TSBUFFRS - Temporary Storage buffers based on SIT TS=.
  - TSGENRAL - Temporary Storage general usage based on SIT TS= and the DFHTEMP CISZ.
  - TSMAIN - Temporary Storage main storage areas based on usage.

CICS Subpools

- Special subpools:
  - KESTK24E and KESTK31E - every CICS nucleus module active in the task execution hierarchy needs an amount of "STACK" storage, and CICS may need to expand what was allocated at initialisation time in its "Stack Extension" subpools; they use Partition GETVIS and not DSA storage, and only DFHPD410 DATA KE=1 and a dump will show their usage.
  - ZCTCTUA - TCTUA storage based on the number of terminals, which will exist in 24-bit or 31-bit subpools according to SIT TCTUALOC=BELOW|ANY, and the DSA used will depend on SIT TCTUAKEY=USER|CICS; I fixed one customer SOS Below by telling them to switch to TCTUALOC=ANY.
- The SM domain control blocks are allocated in Partition GETVIS and not in DSA storage in order to reduce the risk of them being accidentally overlaid.

CICS Storage Requests

- DSA storage is managed by CICS GETMAIN/FREEMAIN.
- Application EXEC CICS GETMAIN/FREEMAIN acquires and frees element storage.
- *Note: CSFE storage freeze means that FREEMAIN does not free storage until end-of-task.*
- CICS and other products will use GETMAIN/FREEMAIN services.
- A GETMAIN will result in CICS SUSPEND (a WAIT) on xDSA or ExDSA if the storage is not available, although specifying NOSUSPEND will result in a NOSTG response.
- A transaction defined with a non-zero DTIMOUT value and SPURGE=YES will be purged if there is a long task xDSA or ExDSA wait.
- Task-related storage is automatically freed at end-of-task.
- However, GETMAIN SHARED storage must be explicitly freed.
- You will see most GETMAIN/FREEMAIN activity with SM level 1 trace active.
- However, Quick-Cell trace activity requires SM level 3 trace to be active, e.g. CETR SM=1-3 or SIT STNTRSM=(1,3).

What is Short-On-Storage (SOS)?

- *The clue is in the name!*
- The DFHSM0131 or DFHSM0133 message is telling you that what you are asking CICS to do means that the available Below (24-bit) or Above (31-bit) storage is not enough for it to confident that it can continue without a problem.
- CICS either hasn't the necessary extents to fulfil a request, or thinks that it might need an available extent in the future and there aren't any.
- Before it got to this state, CICS will have noticed that storage usage was starting to show signs of "stress", and will have been proactively trying to free some unused storage, e.g.
  - Program Compression is used to progressively delete unused programs.
  - CICS will lower the priority of new tasks to reduce the possibility of them adding to the problem while helping older tasks to terminate more quickly and free their storage.
- At SOS, CICS will not allow new tasks to be attached.
- If CICS can recover, it will tell you with a DFHSM0132 or a DFHSM0134 message.
- You may end up with CICS going in and out of SOS a number of times, and it may hang.

The SIT MXT parameter and SOS

- MXT determines how many *user* tasks can be under the control of the CICS Dispatcher at any one time - CICS system tasks are not subject to MXT.
- At MXT, new tasks will be suspended in an MXT wait until other user tasks terminate.
- Setting MXT lower than you might want trades MXT wait time for SOS, and may be the only way to resolve DSALIM issues, however, you should aim to fix the underlying problem.
- The larger the MXT value, the more storage that can be used at any one time and the more likely that SOS will occur if DSALIM or EDSALIM are not sized to match it.
- If you want a simple way to size (E)DSALIM, try to use sets of CICS Statistics to get "normal" values for what you see being used:
  - Required DSALIM = (MXT/Normal Peak MXT) * Normal Peak DSALIM.
  - Do the same for EDSALIM.
- *Note: A user transaction is subject to any TCLASS limit before the MXT limit.*

Can I get do anything to work around SOS?

- If you keep a CEMT task running permanently, you can take some action while CICS is actually in the SOS state.
- The main problem is that you often don't know what the root cause is.
- Possible actions:
  - Using CEMT I DSA to add (E)DSALIM storage may provide relief.
  - Using CEMT I TA to purge one or more tasks may provide relief, but it does not identify the root cause - a task in (E)xDSA wait may be the victim.
  - Use CEMT I SYS to reduce MXT.
- Vendor transaction displays show more about the tasks and even the CICS system as a whole, so you might have more information to help you, but trying to start a new Vendor transaction inside the CICS at SOS won't work!

Could I have prevented SOS?

- Regularly track peak (E)DSALIM and MXT using DFH0STAT, DFHSTUP or a Vendor product so that you know what is "normal".
- Trigger a warning when peak usage exceeds something like 80% of maximum, and when it triggers, try to identify the cause(s) and make changes if that is appropriate.
- However, a rogue application change, a bug or an capacity issue can catch you by surprise.
- The next three slides show DFH0STAT output produced *after* CICS had been SOS several times; a red highlight shows data values that may be of interest.
- In terms of the SOS conditions, we see the CICS system a long time after it happened and it does not appear to be under any stress now - statistics alone normally won't give us enough information to tell us what the root cause of SOS is.
- Adding 512K (the difference between DSALIM and the sum of 4 Peak DSA values) may be what is required to avoid SOS, but that would depend on having more than 512K in the GETVIS "largest free area", which we do not have.
- *Note: DFH0STAT shows actual 31-bit GETVIS storage and NOT the "ANY" 24-bit plus 31-bit values shown by GETVIS AR command output.*

Could I have prevented SOS?

```
Partition size established from ALLOC parameter . . :     122,879K

 Storage BELOW 16MB

   Partition GETVIS area size under 16 Mb  . . . . . . :      11,260K
     Partition GETVIS used area below 16 Mb  . . . . . :      11,032K
     Partition GETVIS free area below 16 Mb  . . . . . :         228K
     Partition GETVIS maximum used below 16 Mb . . . . :      11,048K
     Partition GETVIS largest free area  below 16 Mb . :         216K
```

. . . continued on the next slide

Could I have prevented SOS?

```
 Current DSA Limit . . . . . . . :       9,216K        ← 9MB
 Current Allocation for DSAs . . :       9,216K
 Peak Allocation for DSAs. . . . :       9,216K
                                              CDSA             UDSA             SDSA             RDSA             Totals
 Current DSA Size. . . . . . . . :           4,352K           2,560K           1,792K             512K             9,216K
 Current DSA Used. . . . . . . . :           2,580K              28K           1,656K             456K             4,720K
 Current DSA Used as % of DSA. . :              59%               1%              92%              89%               51% of DSA Size
*Peak DSA Used . . . . . . . . . :           4,144K           2,496K           1,744K             460K
 Peak DSA Size . . . . . . . . . :           4,352K           3,072K           1,792K             512K     ← The total is 9.5MB
 Cushion Size. . . . . . . . . . :              64K              64K              64K              64K
 Free Storage (inc. Cushion) . . :           1,772K           2,532K             136K              56K
*Peak Free Storage . . . . . . . :           1,812K           2,532K             256K             288K
*Lowest Free Storage . . . . . . :             208K              64K              48K              52K
 Largest Free Area . . . . . . . :             256K             256K              48K              32K
 Largest Free Area as % of DSA :                5%              10%               2%               6%
 Largest Free/Free Storage . . . :             0.14             0.10             0.35             0.57
 Current number of extents . . . :               17               10                6                2                35
 Number of extents added . . . . :               20               13                6                2
 Number of extents released. . . :                3                3                0                0
 Getmain Requests. . . . . . . . :        3,546,391       41,640,453              831               25
 Freemain Requests . . . . . . . :        3,545,994       41,640,445              727                3
 Current number of Subpools. . . :               39               11                6                4                60
 Add Subpool Requests. . . . . . :          293,675          293,647                6                4
 Delete Subpool Requests . . . . :          293,636          293,636                0                0
 Times no storage returned . . . :                0                0                0                0
 Times request suspended . . . . :               24               54                0                0
 Current requests suspended. . . :                0                0                0                0
 Peak requests suspended . . . . :                4                8                0                0
 Requests purged while waiting :                  0                2                0                0
 Times Cushion released. . . . . :              349              215                0              832
 Times Short-On-Storage. . . . . :                6               15                0               19
 Total time Short-On-Storage . . :   00:01:54.40254   00:00:50.97460   00:00:00.00000   00:02:02.41115
 Average Short-On-Storage time . :   00:00:19.06708   00:00:03.39831   00:00:00.00000   00:00:06.44269
```

Could I have prevented SOS?

```
Loader

  Library Load requests. . . . . . . . . . . . . . . :          1,575  Total Program Uses . . . . . . . . . . . . . . . :      1,678,077
  Total Library Load time. . . . . . . . . . . . . . :  00:00:23.25243  Program Use to Load Ratio. . . . . . . . . . . . :          65.44
  Average Library Load time. . . . . . . . . . . . . :  00:00:00.01475
  Library Load requests that waited. . . . . . . . . :              2
  Total Library Load request wait time . . . . . . . :  00:00:00.03964
  Average Library Load request wait time . . . . . . :  00:00:00.01982
  Current Waiting Library Load requests. . . . . . . :              0
  Peak Waiting Library Load requests . . . . . . . . :              1
  Times at Peak. . . . . . . . . . . . . . . . . . . :              2  Average Not-In-Use program size. . . . . . . . . :            23K
  CDSA                                                                  ECDSA
  ____                                                                  _____
  Programs Removed by compression. . . . . . . . . . :             30  Programs Removed by compression. . . . . . . . . :              0
  Time on the Not-In-Use Queue . . . . . . . . . . . :  16:37:44.18565  Time on the Not-In-Use Queue . . . . . . . . . . :  00:00:00.00000
  Average Time on the Not-In-Use Queue . . . . . . . :  00:04:55.17636  Average Time on the Not-In-Use Queue . . . . . . :  00:00:00.00000
  Programs Reclaimed from the Not-In-Use Queue . . . :          1,184  Programs Reclaimed from the Not-In-Use Queue . . :         20,886
  Programs Loaded - now on the Not-In-Use Queue. . . :              6  Programs Loaded - now on the Not-In-Use Queue. . :             17
  SDSA                                                                  ESDSA
  ____                                                                  _____
  Programs Removed by compression. . . . . . . . . . :            298  Programs Removed by compression. . . . . . . . . :              0
  Time on the Not-In-Use Queue . . . . . . . . . . . :  15:39:38.02778  Time on the Not-In-Use Queue . . . . . . . . . . :  00:00:00.00000
  Average Time on the Not-In-Use Queue . . . . . . . :  00:00:41.41425  Average Time on the Not-In-Use Queue . . . . . . :  00:00:00.00000
  Programs Reclaimed from the Not-In-Use Queue . . . :      1,230,968  Programs Reclaimed from the Not-In-Use Queue . . :        334,720
  Programs Loaded - now on the Not-In-Use Queue. . . :              8  Programs Loaded - now on the Not-In-Use Queue. . :            105
  RDSA                                                                  ERDSA
  ____                                                                  _____
  Programs Removed by compression. . . . . . . . . . :              3  Programs Removed by compression. . . . . . . . . :              0
  Time on the Not-In-Use Queue . . . . . . . . . . . :  06:05:34.04108  Time on the Not-In-Use Queue . . . . . . . . . . :  00:00:00.00000
  Average Time on the Not-In-Use Queue . . . . . . . :  02:01:51.34702  Average Time on the Not-In-Use Queue . . . . . . :  00:00:00.00000
  Programs Reclaimed from the Not-In-Use Queue . . . :              0  Programs Reclaimed from the Not-In-Use Queue . . :          2,599
  Programs Loaded - now on the Not-In-Use Queue. . . :              0  Programs Loaded - now on the Not-In-Use Queue. . :             23
```

What can cause SOS?

- Inappropriate configuration options somewhere in the tier of products being used, e.g. using SIT TCTUALOC=BELOW when ANY could be used.
- A workload that CICS cannot process fast enough, pushing MXT past its normal peak and increasing the storage requirements; perhaps CICS or even z/VSE does not have the capacity (cpu/storage/dasd) to cope with the workload at that time.
- An unusual combination of task suspend (wait) states that cause a build up of tasks and their storage - this would also be reflected in a high current/peak MXT value.
- An application bug resulting in more storage being used than expected, e.g. a loop or a poor design that results in a build up of VSAM file backout data, or a very large GETMAIN; it could even be due to an application-generated Storage Leak.
- Bugs in z/VM, z/VSE, Vendor, CICS or other IBM products creating unexpected waits or a Storage Leak. (Although not as SOS issue, I saw a recent z/VM bug stop it dispatching one of the z/VSE Virtual Cpus and causing CICS hangs!)
- A recent change somewhere, which could be anywhere in what is often a very complex environment!

A Storage Leak

- A leak will cause a continuous build up in one or more related types of storage due to unpaired GETMAINs and FREEMAINs.
- However, a large amount of a particular type of storage may be "normal" and not a leak.
- Given enough time, it will always result in an SOS or similar out-of-storage condition.
- Adding more storage will just allow CICS to run longer than it was able to before.
- Fragmentation due to the sizes and patterns of GETMAIN/FREEMAIN may look like a leak, but this is correctly known as "Storage Creep" - I haven't seen this affect CICS yet.
- *Note: You may have leaks already, but not run CICS long enough to cause SOS!*

How do I resolve SOS?

- It depends on what you will find as the root cause, and could include one or more of:
  - Find fixes for IBM or Vendor software.
  - Fix the application program(s).
  - Increase (E)DSALIM, and re-configure the partition and/or z/VSE as required.
  - Decrease MXT.
  - Use TCLASS to limit transactions that use a lot of resource to a maximum number.
  - Tune and/or change configuration options in Vendor products, CICS, z/VSE or z/VM.
  - Provide extra cpu or storage capacity and/or faster dasd.

What diagnostic material do I need?

- You want a dump *at* the time CICS says it is SOS, for this you need to use CEMT S SYD(SM0131) ADD SYS MAX(1) and/or CEMT S SYD(SM0133) ADD SYS MAX(1) or a PLT program to add both system dump codes; a WARM start will retain the setting.
- A dump after CICS has recovered may not be able to identify the root cause accurately.
- Allocate a trace table of at least 4096K and use SIT STNTR=1.
- IBM may ask for extra trace levels to be set if we need the problem to be recreated.
- If CICS is hung, a CANCEL dump should be OK, or take a *synchronous* AR DUMP:
  - SUSPEND xx
  - DUMP xx,0-7FFFFFFF,cuu
  - RESUME xx
  - CANCEL xx,NODUMP
- If *you* want to look at the problem, run INFOANA with:

  CALL DFHPD410 DATA AP=1,DS=1,LD=1,SM=1,TR=3,XM=1

What diagnostic material do I need?

- If the analysis indicates that there might be a performance problem, it may help for you to have detailed task performance data for half an hour or more leading up to the SOS.
- It is also helpful for you to know what is "normal" for the transactions that appear in the task performance data so that you can identify when there are problems and what they are.
- You may need z/VSE and even z/VM performance data.
- If you decide that CICS Service needs to look at it:
  - Open a PMR against CICS and tell us which z/VSE release you are on (SIR SYSTEM output is always welcomed).
  - Add background information about what is "normal" if you know that.
  - CICS L2 will give you FTP instructions.
  - FTP the raw dump in *binary*, with CICS SYSLST and PRINTLOG that includes all related messages in *ASCII* (this applies to any type of CICS problem).
  - Don't FTP DFHPD410 output, we will format the raw dump whichever way we need to.
  - Please *don't* send file formats that you can't look at with e.g. Open Office!

What signs do I look for in a dump or statistics?

- What you will see depends on when the dump was taken relative to the SOS condition.
- The obvious sign in Storage Management SM=1 output is SOS below or above, with a DSA flagged as SOS, but this may be the victim and not the culprit.
- SM=1 output and/or statistics shows the number of NOSTG responses, the number of suspended requests and the number of times DSA cushions were released.
    - Each DSA has a "Storage Cushion", which is the minimum amount of contiguous storage that CICS should keep to avoid SOS.
    - When CICS uses some of this, it is a warning of a potential SOS in the near future.
    - The cushion sizes are 64K for the xDSAs, 0K for the EUDSA, 256K for the ERDSA and 128K for the ECDSA and ESDSA.
- The end of the SM=1 domain dump output will show any tasks waiting for xDSA or ExDSA storage and how much they requested.
- The DS=1 domain output may not show a DSA wait, because SMSY may be trying to free storage in case the failing GETMAIN can be retried.
- Loader Domain statistics provide counts for programs removed by Program Compression.

Analyzing SOS Problems

- I will use examples that are similar to real problems, but with a reduced amount of DFHPD410 output to give you an idea of what to look for.
- This may be an unusual number of task waits, e.g. many tasks in FCCIWAIT for a file, which means they are all waiting for one CI split to complete, or LMQUEUE waits waiting for locked resources; when there is a slowdown, it will often result in more tasks running concurrently, pushing up peak MXT and total storage usage.
- It could be an abnormal amount of storage allocated to a CICS subpool, which could also be due to a slowdown, an application bug or even a storage leak.
- Knowing what is "normal" for your system will help.
- *Note: Being at SOS does not mean that there is no storage available in CICS.*
- *Note: CICS Service have seen many SOS problems and will often spot a potential problem relatively quickly, however, there could be some guesswork as we don't know what is "normal".*

Problem #1

- SOS above occurred several times, CICS recovered, and an SM0133 dump was produced.
- SOS has not occurred before in this CICS system.
- No changes were made recently.
- This is a typical symptom of a capacity issue, but is that what it is?
- XM=1 output shows that CICS is close to MXT.
- *The typical peak MXT for this CICS is approximately 230 compared to the 295 here*.

```
==XM: MXT SUMMARY
   Maximum user tasks (MXT):                  310
   System currently at MXT:                   No
   Current active user tasks:                 295
   Current queued user tasks:                 0
 * Peak active user tasks:                    295
 * Peak queued user tasks:                    0
 * Times at MXT limit:                        0
```

- DFHPD410 XM=1 output shows task information and any TCLASS/MXT waits.

Problem #1

- DS=1 is a *snapshot* of task activity at the time of the dump, but it can provide an interesting insight into what was happening before SOS occurred and possibly the root cause.
- Each task that can be dispatched has a state, and you normally see one of three values:
  - "R" for Running - the task is being dispatched by CICS; in an SM013n dump this will be SMSY. (The SM CICS system task SMSY is used to look for and deal with SOS, and normally runs every 3 minutes to check storage conditions, but every 2 seconds or less when CICS is under stress).
  - "D" for Dispatchable - this is an implied wait for access to the cpu, and seeing many of them waiting for QR could indicate a cpu availability problem
  - "S" means the task is in a Suspend (wait) state, optionally with a purge timeout.
- The CICS Problem Determination Guide Chapter 6 describes each "S" state.
- It may show a "normal" wait, e.g. FCIOWAIT is when you are waiting for VSAM I/O to complete (*but it should be for a short duration)*, or an ICWAIT for "n" seconds.
- It may show contention, e.g. FCCIWAIT says that the task's VSAM I/O is being delayed because an active I/O is performing a CI (and possibly a CA) split; only when this is finished does this task get a chance to retry its VSAM request.

Problem #1

- DS=1 shows CICS System Tasks in normal waits with the SM System Task (AD=SM) running as expected during SOS; the dump time was 12:56:04.
- There are many FCPSWAITs for file MAST001, which is the FILE definition STRINGS wait (not LSR string wait); this could be a bad choice for STRINGS or be due to a slowdown stopping them being released fast enough for another task to use.
- There are a small number of FCIOWAITs - *the one below shows a 2-second I/O wait!*
- No other significant types of wait and only there were only a small number of tasks in a "D" status, so there is no *clear* sign of a cpu problem, but it is only a snapshot after all.

```
KEY FOR SUMMARY
    T  = TYPE OF TASK            S=SYSTEM  N=NON-SYSTEM
    S  = STATE OF TASK           D=DISPATCHABLE  S=SUSPENDED  R=RUNNING   E=RESUMED EARLY
. . .
DS_TOKEN KE_TASK  T S F P TT RESOURCE RESOURCE_NAME    W TIME OF      TIMEOUT           AD      XM_TXN_TOKEN
                             TYPE                        SUSPEND      DUE                               (task#)
00940001 03F1D080 S R                                                                   SM
0112157D 046D8B00 N S P N -  FCPSWAIT MAST001          C 12:56:02.210          -        XM     06D2550000058761C
02161329 0466DB00 N S N N -  FCIOWAIT MAST001          W 12:56:02.216          -        XM     06D2550000058773C
```

Problem #1

- SM=1 summarises the status of CICS DSA storage.
- CICS is at SOS above with no EDSALIM expansion possible.
- 1MB of DSALIM expansion is available (4 * 256K extents).
- *Typical peak EDSALIM usage for this CICS is 90MB.*

```
SM Domain status:                  INITIALISED
   Storage recovery:                  YES
   Storage protection requested:      YES
   Storage protection active:         YES
   Reentrant program option:          PROTECT
   Current DSA limit:                    9216K
   Current DSA total:                    8192K
   Currently SOS below 16M:                 NO
   Current EDSA limit:                    110M
   Current EDSA total:                    110M
   Currently SOS above 16M:                YES
```

Problem #1

- The ECDSA large and is close to SOS, with only the cushion of contiguous storage left.

```
==SM: ECDSA Summary
   Size:                          40960K
   Cushion size:                    128K
   Current free space:             1112K  ( 2%)    (some fragmentation)
 * Lwm free space:                   68K  ( 0%)
 * Hwm free space:                 2212K  ( 5%)
   Largest free area:               128K           (at cushion size)
 * Times nostg returned:              0
 * Times request suspended:           0
   Current suspended:                 0
 * Hwm suspended:                     0
 * Times cushion released:            0
   Currently SOS:                    NO
 * Times went SOS:                    0
 * Time at SOS:            00:00:00.000
```

Problem #1

- The EUDSA is large and is at SOS, with a suspended GETMAIN request.

```
==SM: EUDSA Summary
   Size:                           63488K
   Cushion size:                       0K
   Current free space:               128K  ( 0%)    (no fragmentation)
 * Lwm free space:                   128K  ( 0%)
 * Hwm free space:                  1152K  ( 1%)
   Largest free area:                128K           (maximum contiguous storage)
 * Times nostg returned:                0
 * Times request suspended:             1
   Current suspended:                   1
 * Hwm suspended:                       1
 * Times cushion released:              0
   Currently SOS:                     YES
 * Times went SOS:                      1
 * Time at SOS:              00:00:00.000
```

Problem #1

- Although not relevant to this SOS analysis, SM=1 provides an extent summary.
- The EUDSA has 57 extents and only the last one has free storage, which shows no obvious signs of fragmentation.

```
Number of extents:                    57

Extent list:       Start       End           Size      Free

                   06300000    063FFFFF     1024K        0K
                   06800000    069FFFFF     2048K        0K
                   . . .
                   0A800000    0A8FFFFF     1024K        0K
                   0A900000    0A9FFFFF     1024K        0K
                   0AA00000    0AAFFFFF     1024K        0K
                   0AB00000    0ABFFFFF     1024K        0K
                   0AC00000    0ACFFFFF     1024K      128K
```

Problem #1

```
==SM: ESDSA Summary
   Size:                            4096K     (insignificant)
   Cushion size:                     128K
   Current free space:              2620K  (63%)
 * Lwm free space:                   336K  ( 8%)
 * Hwm free space:                  2620K  (63%)
   Largest free area:                772K
 * Times nostg returned:               0
 * Times request suspended:            0
   Current suspended:                  0
 * Hwm suspended:                      0
 * Times cushion released:             0
   Currently SOS:                     NO
 * Times went SOS:                     0
 * Time at SOS:               00:00:00.000
```

Problem #1

```
==SM: ERDSA Summary
   Size:                            4096K      (insignificant)
   Cushion size:                     256K
   Current free space:               312K  ( 7%)
 * Lwm free space:                   312K  ( 7%)
 * Hwm free space:                  2628K  (64%)
   Largest free area:                312K
 * Times nostg returned:                0
 * Times request suspended:             0
   Current suspended:                   0
 * Hwm suspended:                       0
 * Times cushion released:              0
   Currently SOS:                      NO
 * Times went SOS:                      0
 * Time at SOS:              00:00:00.000
```

Problem #1

- An analysis of the task subpools showed that tasks 38 and 141 use a lot of storage, the XM domain has the transaction ids and the usage will need to be validated.
- You can see the difference between element storage (actual usage) and the page storage 4K or 64K multiples suballocated from the DSA extent storage.
- Other user tasks show EUDSA usage from 64K to 320K.

```
==SM: Task subpool summary
   Current number of tasks:     303       (includes system tasks)

   SMX Addr Name     Id Loc Acc   Gets  Frees  Elems  Elemstg Pagestg
    . . .
   03E8A2C4 M0000038 01  B   C       1      1      0        0      0K  (CDSA)
            C0000038 03  A   C       0      0      0        0      0K  (ECDSA)
            B0000038 02  B   U       6      4      2     1584      4K  (UDSA)
            U0000038 04  A   U     130     94     36  1432128   1472K  (EUDSA)
            (task 38)                                (1398.5K)
```

Problem #1

```
03E8A498 M0000141 01  B   C       0      0      0        0      0K
         C0000141 03  A   C       1      0      1       48      4K
         B0000141 02  B   U     520    518      2     9056     12K
         U0000141 04  A   U   62317  62312      5  5085616   5056K
. . .
03E8DB3C M0058770 01  B   C       0      0      0        0      0K
         C0058770 03  A   C       0      0      0        0      0K
         B0058770 02  B   U       1      1      0        0      0K
         U0058770 04  A   U       2      0      2    15680     64K
. . .      (58770 is having the issue with EUDSA storage)
03E8BE64 M0085812 01  B   C       0      0      0        0      0K
         C0085812 03  A   C       0      0      0        0      0K
         B0085812 02  B   U      18     16      2     9056     12K
         U0085812 04  A   U       8      5      3   151632    256K
      (task 85812 is an example of a "normal" task in terms of EUDSA usage)
```

Problem #1

- The CICS subpools show that DFHTDG31 (Transient Data general storage) and DFHTDIOB (Transient Data I/O buffers) use more than 50% of the 40MB of ECDSA storage.
- This could be "normal", but the SIT TD= parameter may be over-allocated.
- No other CICS subpool usage is significant.

```
==SM: Domain subpool summary (ECDSA)
   Name     Id Chn  Initf Bndry Fxlen Q-c   Gets  Frees  Elems  Elemstg Pagestg
   AITM_TAB B3         4K     8   592  Y      60      0     60    35520     40K
   AP_AFCTE C7  Y      4K    16              286      0    286     9968     12K
   AP_TCA31 4D       128K   128  1536  Y   27032  26750    282   433152    564K
   . . .
   DFHTDG31 81               16             6006      0   6006   578544    568K
   DFHTDIOB 84               16                1      0      1 24576000  24000K  (23MB+)
   DFHTDWCB 85         4K    16    64  Y   77707  77707      0        0      4K
```

Problem #1

- DSA waits are shown at the end of SM=1.
- A request for 180912 bytes (177K), which will not fit in the 128K of contiguous storage that is available, and will probably require 192K or 3 * 64K.
- If 192K was added to the existing 64K for task 58770, it would result in 256K, which is a size often seen for transactions in this CICS system and appears to be reasonable.

```
==SM: Suspend queue summary

   KE Task  Tran #  Susptok  Subpool  DSA       Request

   04416780 0058770 0416139B U0058770 EUDSA      180912
```

Problem #1

- The trace shows more than 1,000 FCIOWAITs in a short time.
- A Rexx program analyzed the trace and showed that CICS was dispatched more than 80% of the time, a level of activity that could definitely be causing a slowdown.
- These trace entries lead up to the SOS.
- The X'2C298' byte request is for LE RUWA 31-bit working storage, and CICS was looking for additional 64K subpool pages.

```
AP 1940 APLI  ENTRY - FUNCTION(START_PROGRAM) PROGRAM(NPG00001) CEDF_STATUS(CEDF) EXECUTION_SET(FULLAPI) ENVIRONMENT_TYPE(EXEC)
                      SYNCONRETURN(NO) LANGUAGE_BLOCK(06791648) COMMAREA(00000000 , 00000000) LINK_LEVEL(1) SYSEIB_REQUEST(NO)

              TASK-58770 KE_NUM-0020 TCB-004E5000 RET-8D37F904 TIME-12:56:02.2605341901 INTERVAL-00.0000003125    =098282=
                1-0000  00580000 000000DA 00000000 00000000  B81B5D00 00000000 02680100 D6E5E2C2  *.................).........OVSB*
                  0020  E4F2F040 064EA730 B5000000 064EA730  01000001 01F10202 07F11D20 06791648  *U20 .+x......+x......1...1......*
                  0040  00003928 00000000 00000000 00000000  00010002 02000000                    *.......................        *

SM 0301 SMGF  ENTRY - FUNCTION(GETMAIN) GET_LENGTH(2C298) SUSPEND(YES) REMARK(RUWAPOOL) STORAGE_CLASS(TASK31)

              TASK-58770 KE_NUM-0020 TCB-004E5000 RET-8D3187B8 TIME-12:56:02.2605344401 INTERVAL-00.0000002500    =098283=
                1-0000  00400000 0000000E 00000000 00000000  B6580000 00000000 01400102 00000000  *. ...................... ......*
                  0020  00000000 19400000 0002C298 00010058  00000000 0100D9E4 E6C1D7D6 D6D307C0  *..... ....Bq..........RUWAPOOL..*
```

Problem #1

```
SM 1206 SMPQ  *EXC* - Insufficient_storage_to_satisfy_request - FUNCTION(ALLOCATE_PAGEPOOL_STORAGE) SUBPOOL_TOKEN(03089E30)
                   GET_LENGTH(2C2B0) SUSPEND(YES)

              TASK-58770 KE_NUM-0020 TCB-004E5000 RET-8B8D4714 TIME-12:56:02.2605378776 INTERVAL-00.0000034375    =098284=
                1-0000  00380000 0000010A 00000000 00000000  BEF00000 00000000 01000201 03089E30  *................0.............*
                  0020  0002C2B0 03012332 0441DF10 00000040  0B882284 018822B4                    *..B............ .h.d.h..        *
                2-0000  E4F0F0F5 F8F7F7F0                                                         *U0058770                        *
                3-0000  C5E4C4E2 C1404040                                                         *EUDSA                           *

SM 1001 SMSQ  ENTRY - FUNCTION(SUSPEND_REQUEST) GET_LENGTH(2C2B0) SUBPOOL_TOKEN(03089E30) RETRY(NO)

              TASK-58770 KE_NUM-0020 TCB-004E5000 RET-8B8D65D6 TIME-12:56:02.2605390026 INTERVAL-00.0000011250    =098285=
                1-0000  00300000 00000090 00000000 00000000  BC800000 00000000 02000101 0002C2B0  *............................B.*
                  0020  03089E30 03012332 0441DF10 02000040                                       *...............                 *
. . . . .
SM 080A SMSY  *EXC* - Short_on_storage_in_the EUDSA

              TASK-SM     KE_NUM-001D TCB-004E5000 RET-8B88A816 TIME-12:56:02.2605576276 INTERVAL-00.0000002500    =098321=
                1-0000  C5E4C4E2 C1404040                                                         *EUDSA                           *
                2-0000  00020000                                                                  *....                            *
                3-0000  00000000                                                                  *....                            *
                4-0000  00000001                                                                  *....                            *
```

Problem #1 Conclusion

- The large task and TD storage usage was "normal".
- Nothing else was observed to be "abnormal" in this CICS system.
- No changes were made before the SOS occurred.
- SOS is not a common problem.
- This is a match for a short-term capacity problem.
- One option is to increase EDSALIM.
  - The typical peak MXT is 230 and peak EDSALIM usage is 90MB, therefore the suggested new EDSALIM value is (310/230) * 90MB = 121MB.
  - With 1MB still available in DSALIM, SOS below is unlikely to occur.
- Another option would be to reduce MXT to a bit less than 295.
- However, I would suggest a review of the performance of the z/VSE and CICS system to ensure that it has the capacity to handle the workload during times of peak load with appropriate response times.

Problem #2

- DSA usage grows quickly in size until an SOS below occurs and CICS does not recover.
- This happens every time CICS is started.
- This is the classic symptom of a Storage Leak, and should be easy to diagnose.
- Being at MXT may or may not be significant.

```
==XM: MXT SUMMARY

   Maximum user tasks (MXT):                  60
   System currently at MXT:                   Yes
   Current active user tasks:                 60
   Current queued user tasks:                 3
 * Peak active user tasks:                    50
 * Peak queued user tasks:                    6
 * Times at MXT limit:                        4
```

- DS=1 shows no obvious sign of any problems, so I will not include any output from it.

Problem #2

- SM=1 shows SOS Below.

```
===SM: STORAGE MANAGER DOMAIN - SUMMARY

   SM Domain status:                  INITIALISED
   Storage recovery:                  NO
   Storage protection requested:      YES
   Storage protection active:         YES
   Reentrant program option:          PROTECT
   Current DSA limit:                    7424K
   Current DSA total:                    7424K
   Currently SOS below 16M:                YES

   Current EDSA limit:                     55M
   Current EDSA total:                     40M
   Currently SOS above 16M:                 NO
```

Problem #2

```
==SM: UDSA Summary

   Size:                         512K
   Cushion size:                  64K
   Current free space:           212K  (41%)
 * Lwm free space:               140K  (27%)
 * Hwm free space:               496K  (96%)
   Largest free area:            152K
 * Times nostg returned:            0
 * Times request suspended:         0
   Current suspended:               0
 * Hwm suspended:                   0
 * Times cushion released:          0
   Currently SOS:                  NO
 * Times went SOS:                  0
```

Problem #2

```
==SM: CDSA Summary

   Size:                           1280K
   Cushion size:                     64K
   Current free space:              688K  (53%)
 * Lwm free space:                  120K  ( 9%)
 * Hwm free space:                  688K  (53%)
   Largest free area:               240K
 * Times nostg returned:               0
 * Times request suspended:            0
   Current suspended:                  0
 * Hwm suspended:                      0
 * Times cushion released:             0
   Currently SOS:                     NO
 * Times went SOS:                     0
```

Problem #2

```
==SM: SDSA Summary

   Size:                            5120K     (the biggest usage)
   Cushion size:                      64K
   Current free space:                60K  ( 1%)
 * Lwm free space:                    60K  ( 1%)
 * Hwm free space:                   256K  ( 5%)
   Largest free area:                 60K
 * Times nostg returned:                0
 * Times request suspended:             0
   Current suspended:                   0
 * Hwm suspended:                       0
 * Times cushion released:              8
   Currently SOS:                     YES
 * Times went SOS:                      1
 * Time at SOS:              00:00:00.000
```

Problem #2

```
==SM: RDSA Summary

   Size:                             512K
   Cushion size:                      64K
   Current free space:               216K  (42%)
 * Lwm free space:                   216K  (42%)
 * Hwm free space:                   292K  (57%)
   Largest free area:                216K
 * Times nostg returned:                0
 * Times request suspended:             0
   Current suspended:                   0
 * Hwm suspended:                       0
 * Times cushion released:              0
   Currently SOS:                      NO
 * Times went SOS:                      0
 * Time at SOS:              00:00:00.000
```

Problem #2

- The issue is related to the size of the SDSA.
- Look at the big difference between Gets and Frees and the number of elements, and is a typical sign of a leak!
- GETMAIN SHARED has no task-related information maintained, so who is requesting it?

```
==SM: Domain subpool summary (SDSA)
   Name     Id Chn  Initf Bndry Fxlen Q-c   Gets  Frees  Elems  Elemstg Pagestg
   APECA    5D                8     8          1      1      0        0     0K
   DFHAPU24 46               16                1      0      1     3584     4K
   LDPGM    28               16               20     14      6   100128   108K
   LDRES    24               16                1      0      1    23952    24K
   SMSHRU24 60  Y            16            14087      2  14085  5032704  4924K
```

Problem #2

- Perhaps there is some evidence in the trace, if not, auxtrace might be an option.
- There are more than 300 abbreviated trace entries for GETMAIN SHARED that are identical apart from the task number.
- This one shows the EXEC CICS request for task 14656, it was successful and returned the address EAFF40.

```
14656 1 AP 00E1 EIP   ENTRY GETMAIN                                 0004,05F95B08 .9$.,09000C02 ....          =174462=
. . .
14656 1 SM 0C01 SMMG  ENTRY GETMAIN            154,NO,SHARED_USER24,EXEC                                      =174465=
. . .
14656 1 SM 0C02 SMMG  EXIT  GETMAIN/OK         00EAFF40                                                       =174468=
14656 1 AP 00E1 EIP   EXIT  GETMAIN            OK                   00F4,00000000 ....,00000C02 ....          =174469=
```

Problem #2

- If we look at the full trace, we see more information.

```
AP 00E1 EIP ENTRY GETMAIN                               REQ(0004) FIELD-A(05F95B08 .9$.) FIELD-B(09000C02 ....)

              TASK-14656 KE_NUM-001F TCB-00472000 RET-85B81138 TIME-09:39:15.3141357197 INTERVAL-00.0000003125    =174462=
 . . .        *** The EXEC CICS return address is 05B81138 when you remove the 31-bit addressing 8 bit ***

SM 0C01 SMMG  ENTRY - FUNCTION(GETMAIN) GET_LENGTH(154) SUSPEND(NO) STORAGE_CLASS(SHARED_USER24) CALLER(EXEC)

              TASK-14656 KE_NUM-001F TCB-00472000 RET-851FEFCC TIME-09:39:15.3141391884 INTERVAL-00.0000003437    =174465=
                1-0000  00780000 00000011 00000000 00000000  B6580000 00000000 0208016C 05B81B8E  *.........................%....*
                  0020  05F95B70 0508FB6C 00000154 80000000  05B81B8E 027F2001 007F0BD2 00000000  *.9$....%............"...".K....*
                  0040  0508F988 0508F680 85220FC4 050657A8  00000000 05221FC4 0508FB6C 00000004  *..9h..6.e..D...y.......D...%....*
                  0060  00000000 80000080 44040140 07010000  00680000 00000028                    *.......... ...........        *
. . .
SM 0C02 SMMG  EXIT  - FUNCTION(GETMAIN) RESPONSE(OK) ADDRESS(00EAFF40)

              TASK-14656 KE_NUM-001F TCB-00472000 RET-851FEFCC TIME-09:39:15.3141476572 INTERVAL-00.0000000937    =174468=
                1-0000  00780000 00000011 00000000 00000000  B6580000 00000000 0208016C 05B81B8E  *.........................%....*
                  0020  05F95B70 00EAFF40 00000154 80000000  05B81B8E 027F2001 007F0BD2 00000000  *.9$.... ............"...".K....*
                  0040  0508F988 0508F680 85220FC4 050657A8  00000000 05221FC4 0508FB6C 00000004  *..9h..6.e..D...y.......D...%....*
                  0060  00000000 80000080 44040140 07010000  00680000 00000028                    *.......... ...........        *
```

Problem #2

- This is what I see when I use an internal IBM dump browser and go backwards from the EXEC CICS return address -2 to look for a program eye-catcher.

```
05B80EA8    -290   C4C6C8E8 C9F4F1F1 58F00014 58F0F0B4 | DFHYI411.0...00. |
05B80EB8    -280   58F0F00C 58FF000C 07FF0000 00000000 | .00............. |
05B80EC8    -270   47F0F028 00C3C5C5 00000000 00000014 | .00..CEE........ |   (LE program)
05B80ED8    -260   47F0F001 4ACEAC00 05B8790C 00000000 | .00.¢........... |
05B80EE8    -250   00000000 00000000 90ECD00C 4110F038 | ..............0. |
05B80EF8    -240   98EFF04C 07FF0000 05B87860 05B87954 | q.0<.......-.... |
05B80F08    -230   05B8A330 05B878C0 05B87860 05B87E20 | ..t........-..=. |
05B80F18    -220   05B8AAD0 05B87920 00000000 00000008 | ................ |
05B80F28    -210   C3C9C3E2 D6E2E4D4 F2F0F1F4 F0F7F1F9 | CICSPG0120140719 | ← the actual program
05B80F38    -200   F1F0F2F2 F1F8F0F1 F0F1F0F1 00000000 | 102218010101.... |
. . .
05B81118     -20   01544110 D06841E0 80AE41F0 D1704100 | ...........0J... |
05B81128     -10   805290E0 10009680 100858F0 CF8405EF | ......o....0.d.. |   (the EXEC CICS call)
```

Problem #2

- The same return address can be found in the other trace entries for SHARED-24.
- I can also see the program in the LD=1 output shown below.
- Using the return address 05B81138, find the next higher Load Point and go back one program.
- If the return address is not found in LD=1 OUTPUT, you will need to use eye-catcher information to identify whose code it is.

```
PROGRAM STORAGE MAP

PGM NAME ENTRY PT  CSECT   LOAD PT. REL. PTF LVL. LAST COMPILED  COPY NO. USERS     LOCN  TYP

CICSPG01 85B80EC8 DFHYI411 05B80EA8 411                              1        11     ESDSA RPL
CICSPG05 85B817E8 DFHYI411 05B817C8 411                              1         2     ESDSA RPL
```

Problem #2 Conclusion

- It is a leak of 154-byte requests in the SDSA.
- Program CICSPG01 is performing a GETMAIN SHARED but there appears to be no code that is issuing a FREEMAIN.
- Something needs to be fixed in the application.

Problem #3

- There will not be time to look at this problem's dump information, but I have provided it as an example of the kind of symptoms that may be seen and how a different type of SOS problem would be handled by CICS Service.
- After a CICS/VSE 2.3 and a z/VSE migration (with testing), recoverable SOS conditions started to occur and SM0131 SOS Below dumps were produced.
- SM=1 showed DSALIM=8192K, the UDSA had 4.75MB with 326K contiguous free, the CDSA had 1.5MB with 244K contiguous free, *the SDSA had 1.25MB with 44K contiguous free and was SOS due to the 64K cushion having been released*, the RDSA had 0.5MB with 148K contiguous free.
- XM showed MXT=150 user tasks, current active 139 with peak active 149.
- These symptoms suggest a capacity problem, and GETVIS availability suggested that it would not be difficult to add another 256K or 512K to DSALIM.
- But a migration was also involved, and making a diagnosis based only on symptoms can be very dangerous, so we needed to do more analysis.

Problem #3

- The DS domain showed:
  - More than 50 FCCIWAIT for the same file, with one task in FCIOWAIT and a few FCPSWAITs - the culprit or a victim? (Trace analysis shows it is the victim.)
  - More than 40 tasks in a Dispatchable state - a significant backlog of tasks that could be running - it is cpu availability or something else? (Trace analysis shows the cause.)
  - Other task states are "normal".
- A trace analysis showed that CICS was *very* busy (*and* an unusual amount of elapsed time was captured), that could explain why there were Dispatchable states, but is more than 40 of 139 tasks reasonable? (My experience would say "no".)

```
Trace elapsed time 35.0872540312   (seconds)
Task dispatch time 35.0501192200
Task idle time      0.0371348112
Task elapsed utilisation 99.89%
```

Problem #3

- The analysis showed that some tasks ran quickly and completed, but others ran very slowly, and two of the tasks with low task numbers (i.e. they started before many of the others in the reported interval) were dispatched for more than 34 seconds and did not even finish.

```
Task 32009 *** Response 35.0872540312 Total Dispatched 16.8256172195 Total Wait 18.2616368117 Elapsed:Dispatch ratio = 2.09
Task 32077 *** Response 35.0816723125 Total Dispatched 17.6481242812 Total Wait 17.4335480313 Elapsed:Dispatch ratio = 1.99
```

- The execution summary is not "normal", and GEMAST is not the file with the FCCIWAIT states; here is a part of the task summary showing some very long times.

```
Task 32077 Dispatched Elapsed: 0.2544422187 Start: =003323= 10:18:56.5282236262 End: =003425= 10:18:56.7826658449
Task 32077 Wait       Elapsed: 0.0026335313 Possible dispatch delay  (high priority TCP task is dispatched)
Task 32077 Dispatched Elapsed: 0.2656721875 Start: =003801= 10:18:56.7852993762 End: =003903= 10:18:57.0509715637
Task 32077 Wait       Elapsed: 0.0356217500 Possible dispatch delay  (so many VSAM requests that CICS let another task run)
Task 32077 Dispatched Elapsed: 0.2733920937 Start: =005394= 10:18:57.0865933137 End: =005496= 10:18:57.3599854074
Task 32077 Wait       Elapsed: 0.2659688125 Possible dispatch delay  (same as the previous reason)
Task 32077 Dispatched Elapsed: 0.2625850938 Start: =005914= 10:18:57.6259542199 End: =006016= 10:18:57.8885393137
Task 32077 Wait       Elapsed: 0.2664264375 Possible dispatch delay  (same as the previous reason)
Task 32077 Dispatched Elapsed: 0.0000440000 Start: =006482= 10:18:58.1549657512 End: =006484= 10:18:58.1550097512
Task 32077 Wait       Elapsed: 0.5313980000 FUNCTION(WAIT_OLDW) RESOURCE_NAME(GEMAST) RESOURCE_TYPE(FCIOWAIT)
```

Problem #3

- The full trace showed more than 13,000 VSAM exception *EXC* trace entries.
- The exception entry was repeated in the long dispatch times when the start and end trace sequence number are used to view the full trace output.

```
AP 04B7 FCVS *EXC* VSAM EXCEPTION - VSAM RPL
            TASK-32077 KE_NUM-0079 TCB-0031A000 RET-8D49EDE2 TIME-10:18:54.4216879074 INTERVAL-00.0000030937    =003324=
             1-0000  00780000 00000038 00000000 00000000  B4278C3C 00000000 05000100 00000000  *................................*
               0020  00000000 04D157B0 00000009 00000000  00000000 00000000 00000000 00000000  *.....J..........................*
               0040  04D157B0 00000000 0677C2E0 0677C2E0  00000092 04D1CB70 00000000 00000000  *.J........B...B....k.J..........*
               0060  00140000 00000000 00000102 02010000  02000012 00000000                    *........................        *
             2-0000  0011003C 00000000 0677C2E0 0677C2E0  00000092 00000004 0057B210 07100000  *..........B...B....k............*
               0020  98100000 60080014 00000000 00000000  00003710 0677D4B8 00800000           *q...-.................M.....    *
```

- The second data area is the RPL, and offset X'24' contains the return code and error code.
- z/VSE Messages Volume 2 says X'08' and X'14' is a VSAM CI exclusive control issue, and is significant when found in almost every exception trace.
- If you saw something like X'08' and X'10', it is a No Record Found, which is a "normal" exception in most cases (unless it repeats in a program loop).

Problem #3 Conclusion

- CICS and VSAM are looping on an exclusive control conflict until it is resolved and the I/O can actually be started.
- This is not how they are designed to work together, so we would work with VSAM Service.
- One or more tasks monopolising CICS will have an impact on its ability to run normally, there will be a build up of tasks and their storage requirements until it is resolved, and even then it will take time to get back to normal as CICS will probably have a backlog of work to deal with.
- Interestingly, CICS detects when a task performs many consecutive VSAM requests to the same file, and does a CHANGE_PRIORITY to recalculate (i.e. lower) its dispatch priority and allow other user tasks to do some work.
- This will also affect lower PRTY z/VSE partitions while it is happening.
- CICS Monitoring task performance records would show a large ratio of VSAM requests to application EXEC CICS requests in this situation - look at my WAVV presentations to see this and see how a different (but long-since fixed) VSAM bug affected performance and why.

Thank You

Please forward your questions or remarks to
PoilMike@uk.ibm.com
zvse@de.ibm.com

z/VSE Live Virtual Classes

z/VSE	@ http://www.ibm.com/zvse/education/
LINUX + z/VM + z/VSE	@ http://www.vm.ibm.com/education/lvc/

Read about upcoming LVCs on @ http://twitter.com/IBMzVSE

Join the LVC distribution list by sending a short mail to zvse@de.ibm.com