

Controlling Transaction Input into CICS/TS

September 11, 2006

By: Eugene S. Hudders

CICS/TS Performance Tuning Using C\TREK Course

This seminar covers a lot of tuning material and will require 8 hours a day. The total time for the seminar is 40 hours.

The followings topics included in this course are:

- Introduction to CICS Performance Tuning
- Using Operating System Information to Tune CICS/TS
- Tuning CICS/TS Processor Cycles
- Tuning Real Storage in CICS/TS
- Tuning Virtual Storage in CICS/TS
- Tuning CICS/TS Transaction Controls
- Tuning On Line VSAM Files
- Tuning NSR Files in CICS/TS
- Tuning CICS/TS LSR Buffer
- Tuning the Index CISZ
- Reviewing VSAM Free Space
- Reviewing the DB2 Interface

If you wish to attend this course, please send us an email at ctrek@actpr.com

Controlling the number of transactions entering into CICS/TS is an important tuning area. Allowing a larger volume than can be handled by CICS/TS can result in Short on Storage (SOS) conditions and create a lack of other resources such as strings. However, reaching the maximum task limit can also have the negative effect of creating an “artificial bottleneck” when resources are available. In both cases, response times are negatively affected. An artificial bottleneck is one that causes a delay when there are sufficient resources available to handle the tasks. This article reviews two important user controls available to control the transaction flow into CICS/TS and possible causes for the MXT limit to be reached. The two areas that will be reviewed in this article are the MXT and TCLASS parameters. This article briefly mentions the effect of MXT on the Workload Manager (WLM) but if more details are desired, then you should refer to another article called “z/OS Workload Manager and CICS/TS MXT” available at www.ctrekcorp.com.

The first parameter available to control transaction flow into CICS/TS is the maximum task parameter (MXT) found in the System Initialization Table (SIT). The user can specify a value from minimum of one (1) task to a maximum of nine hundred and ninety nine (999). The default value is five (5). The selected MXT value is used to control user tasks, that is, system tasks such as CSSY and CSTP are not included in the control. It is also important to note that a system task is not one that begins with the letter “C” as is sometimes erroneously defined. For example, CEMT and CEDA are considered a user or non-system tasks (Figure 1). System tasks are specifically designated by CICS/TS when the task is attached. The system will attach user tasks up to the MXT value. Excess tasks above the MXT value are not attached until the number of attached tasks falls below the MXT value. There is no

mechanism available to establish a threshold point at which the system will automatically purge pending MXT transactions in excess of a defined threshold as there is with the TCLASS definitions. The initial MXT value specified in the SIT can be overridden in the CICS/TS start-up or dynamically via a CEMT command.

```

GTAS ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      1DD6EC00  DATE 09/18/2006
APPLID CICSTS31                  KERNEL DOMAIN      TIME 11:23:55
VERSION 6.4                      ACTIVE TASKS       TERM 0205

  A IND KE_IND  TRAN  MODE  TRANNUM  TERMIID  PROGRAM  WAIT  STATE  STATUS  TYPE
- 23  1D      QR    QR      21      DFHFCQT  MVS  SUSP  NOT RUNNING  SY
- 24  1E      CFQR  QR      26      DFHSHSY  MVS  SUSP  NOT RUNNING  SY
- 25  1F      CSHQ  QR      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 29  26      QR    QR      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 30  27      CSZI  SZ      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 34  2D      SP    SP      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 35  2F      SO    SO      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 36  30      QR    QR      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 37  31      QR    QR      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 38  32      QR    QR      22      DFHSZRM  MVS  SUSP  NOT RUNNING  SY
- 125 8A      TREC  QR     386     0205     KVPGTAS  MVS  SUSP  NOT RUNNING  SY
- 126 8B      TSVR  QR     35      DFHEDAD  MVS  SUSP  NOT RUNNING  SY
- 127 8C      TREK  QR     364     0204     KVPDTAD  MVS  SUSP  NOT RUNNING  SY
- 128 8D      CEDA  QR     396     0214     DFHEDAD  MVS  SUSP  NOT RUNNING  SY
- 129 8E      CEMT  QR     391     0213     DFHEMTD  MVS  SUSP  NOT RUNNING  SY

ENTER REFR ENTER+CSR SPECIFIC TASK PF1 HELP PF3 PREV PG PF5 MEMORY
PF7 BACKWARD PF8 FORWARD (P,F,T)IN ACTION+PF12 TERMINATE SPECIFIC TASK CLR MENU
  
```

Figure 1. Transaction Type: SY=System NS=Non-System

MXT is a global system control value, that is, it is based on the total number of attached tasks and not the type of transaction being attached. In other words, MXT controls the total number of tasks that can be in CICS/TS at any one time but cannot discern what particular transactions are coming into the system. MXT is a control available to the user to control the amount of virtual and real storage that can be allocated at any one time to support task processing. If the MXT figure is set to a higher than required value, then you can be exposed to SOS conditions, wasted CPU cycles as a result of unnecessary WLM scans of the associated unused performance blocks and waste pre-allocated Kernel task storage. However, if the MXT figure is set to a lower than required value, then you can delay the attachment of tasks as a result of reaching the maximum MXT figure established. In either case, you could affect response time. Unused system resources such as CPU cycles are lost, that is, you cannot “save them for a rainy day”.

MXT is usually an inherited number in many installations, that is, that was the value someone established in the SIT many years ago and has been used from one CICS/TS release or version to another. In many installations, the MXT is not analyzed until MXT conditions occur. Selecting a MXT value is not an easy task. There are very few formulas that can be used to establish the value. In many installations, the value is set through a series of iterations until a comfort level is found. In general terms, the measured peak number of tasks should fall around 60 to 70% of the MXT value. The MXT calculation using this percentage range should be calculated over a week of observations. In other words, if you find that over a week the peak number of tasks processed is 50, then the MXT should be set around 70 to 80 tasks (rounded to

a particular multiple such as five). A possible formula that can be used to determine the MXT value is as follows:

$$\text{MXT} = ((\text{Transaction Rate/Second} * 1.5) + (\# \text{ of long running transaction} * 1.25) + (\# \text{ of IIOOP sessions} * 2.5) + (\# \text{ of conversational transactions}) + 25)$$

NOTE 1: If transaction rate/second is less than 1, use 6

NOTE 2: If conversational transactions < 1, use 4

NOTE 3: If MXT results in a production system < 40, use 40

NOTE 4: MXT should be set to a minimum in test systems to reduce WLM overhead

The above formula is useful if you have information about you current system. If you are just starting, the required information is not available, so you may want to start with a value of 40.

As previously mentioned, the MXT value is usually not reviewed until the system starts to incur in MXT conditions that affect response times. The simple solution to the MXT condition is to simply raise the MXT value without reviewing what caused the condition to occur. Unfortunately this “solution” is the one used by many system programmers. There are many causes to an MXT condition, many of which are associated with another performance problem occurring in the system. Raising the MXT value may resolve the MXT situation but may not resolve the actual problem causing the system to reach the MXT limit. If your system was running without any MXT conditions, then the question is: Why is the system going to MXT if it was previously working well? There are many factors that have to be considered some of which are:

- 1) Was there a new or release upgrade made in the system such as new application or feature installed, new system software (e.g., operating system or CICS/TS) or was any maintenance applied?
- 2) Has there been an increase in the number of users on the system as a result of new features; application upgrades company mergers or acquisitions, etc.?
- 3) Are there any bottlenecks in the system such as exclusive control conflicts, short on strings, short on storage conditions (special case discussed later), poor look-aside buffer hit ratios, queued resources, lack of sufficient inter-system communication sessions, bottlenecks in inter-communicated systems (MRO or ISC), etc.?
- 4) Has there been any consolidation of CICS/TS systems in the past few weeks?

Some people simply ignore MXT conditions because they have not reached a certain threshold like 1% to 3% of the total transactions. It is important to note that MXT is simply a warning mechanism used by CICS/TS to alert the user that the system is reaching a limit condition that requires some action to be taken. To ignore this indication without analyzing the cause could result in problems down the line. Sometimes the MXT condition can be attributed to some type

of growth factor. This type of growth factor is usually easy to identify because over a recent period of time you were occasionally experiencing an MXT condition. If historic MXT data can be reviewed, then it would be interesting to note if the MXT condition was caused by growth over time. You would be able to see the peak growing across time. If it was due to growth, then a historical growth rate could be used to adjust the MXT value. If it was caused by increased volume, then the MXT figure has to be adjusted, possibly using the peak as a guide.

MXT is sometimes caused by changes in the operating environment such as new software releases, upgrades or by regular vendor maintenance (e.g., PTF). In this case it is usually very important to contact the vendor immediately to see if the cause has been reported and/or fixed. In some cases it may require backing out the new software maintenance or upgrade. In other cases it may require the use of traces and dumps to determine the cause of the problem. The solution to this type of problem is usually in the vendor's hands (although there are cases where the user is the one that identifies the problem and fix) and your function is to supply information and test any suggested changes. An example of this type occurred when IBM decided to change the default index CI size (CISZ) back a few years ago. Many installations were affected when the upgraded to the new release of z/OS because the default index CISZ was larger than before and very well tuned CICS/TS index LSR buffer pools were not prepared for the change. So, a file that previously had a default index CISZ of 2048 bytes now had a 4096-byte size changing from the 2K to the 4K buffer pool. If there were insufficient 4K buffers to handle the new definition, the look-aside hit ratio went down requiring physical I/O and taking longer to access the desired record. The result was that tasks waited longer in memory and the number of waiting transactions grew resulting in MXT or SOS conditions in some installations.

In many cases the MXT condition is caused by things within one's operating environment. For example, an inter-connected system called AOR1 that communicates with AOR2 may suffer an MXT condition because something is wrong such as MXT or SOS within AOR2. This type of problem is sometimes referred to as a "sympathy sickness", that is, "you get sick and I get sick too!". So, even though AOR1 reflects the MXT condition, the problem resolution is in identifying why AOR2 went sick and fixing that condition. Therefore, it is imperative that the cause of MXT conditions be identified and resolved quickly. Unfortunately, the identification and resolution may take a while. You need to find the clues that identify the problem. For example, inter-system problems may also show waits for sessions between the two systems.

The MXT condition may be caused by things within the region itself. For example, a programming change caused a certain very active file to suffer exclusive control waits and/or string shortages. So, the MXT condition may really be the result of another problem occurring in the system that has to be resolved in order to correct the MXT condition. In other words, if you cannot immediately attribute the reaching of MXT to volume growth, then other possible causes within the region have to be analyzed to determine what caused CICS/TS' dispatch mechanism to slow down and queue more tasks in memory. Some of the areas that should be investigated are:

- 1) Review look-aside hit ratios for both LSR and NSR files – lower hit ratios require more physical I/Os to be performed, thus elongating the task wait time and response time. Add the necessary buffers to resolve this condition.
- 2) Review file conditions – in addition to the look-aside hit ratios, you may want to see if there are exclusive control waits and/or short on string conditions occurring on a file or the LSR pool. These conditions tend to slow down task execution making the task reside in storage longer while new tasks keep flowing into the system. Ensure that there are no volume contentions on the disks that CICS/TS accesses. For example, ensure that someone is not doing volume back-ups during the on-line activity or batch programs are not competing for the on-line files at the same time that CICS/TS is active.
- 3) Review inter-system session activity – verify that the number of sessions between two systems is not the bottleneck. In addition check that the other system does not have other problems that cause a slower response between the inter-connected systems.
- 4) Review Short on Storage (SOS) conditions – another cause of MXT may be a system that enters and leaves the SOS condition. Task attachment is suspended during the SOS period, however, once the SOS condition is resolved, task attachment continues. However, the SOS condition slowed the executing transactions in storage to remain longer. So, if you continue to enter and leave the SOS condition, MXT could occur.
- 5) Review queued resources – the use of ENQ/DEQ is important to protect shared resource use. However, it is just as important to acquire resources in a particular order to avoid deadlock conditions and to release the ENQ as soon as possible to avoid delays. The use of some type of queuing mechanism to protect shared resources in thread safe programs may have an effect on the MXT value if improperly implemented. However, the effect may be mitigated by the extra parallel processing through the use of the Open API TCB structure. Internal CICS locks (e.g., USADLOCK) could also cause a problem that may affect MXT. Lock contentions are usually an IBM problem.
- 6) Review external areas that can affect CICS/TS -- another area that requires analysis is the amount of CPU and real storage resources available to CICS/TS for execution. Lack of real storage will cause CICS/TS to page, slowing down the system sufficiently for tasks to remain longer in the system. The guideline is that the CICS/TS system to have less than 10 page-ins per second. Over commitment of real storage may also be external to CICS such as increasing the DB2 pools or running very large sorts that require a lot of real storage. Also, you need to ensure that the operating system dispatching priority assigned to this CICS/TS through WLM is good and has not changed significantly over the past weeks when the MXT condition appeared. You need to check that other address spaces have not been promoted over the CICS/TS region.
- 7) Review consolidated CICS/TS – installations have been consolidating low volume CICS/TS systems to conserve some resources such as real storage when not using the LPA for the CICS/TS management modules. Review the parameters selected for the consolidation because they may not have been adjusted for the new environment.

Naturally, having a few MXT conditions is not always a cause for alarm or action. For example, you may have 100 MXT situations when processing 250,000 transactions in a day. The MXT condition occurs occasionally at a particular moment in time during the day and goes away for the rest of the day's processing. In these cases, raising the MXT value may increase the number of unused WLM Performance Blocks that have to be scanned generating wasted CPU cycles reflected in the WLM address space. Again, this is part of the analysis that has to be done to determine if a low number of MXT conditions are acceptable.

These are some of the other causes of MXT that should be investigated before raising the MXT value. However, the pressure on the MXT may come from another source such as long running transactions that consume a lot of resources or an excessive number of conversational transactions being present in the system. MXT cannot be used to control the number of a particular type of transaction that comes into the system. Transaction classes (TCLASS) were created for this purpose. Transaction classes are defined in a separate table using RDO. There are some entries supplied by IBM to accommodate the old CMXT (Class Maximum) parameter that had transaction classes in a numbered group of one to ten (1 to 10). The classes are known as DFHTCL01 through DFHTCL10. It should be noted that IBM treats MXT as a special TCLASS called DFHTCL00. All Transactions not assigned to a particular TCLASS are assigned by default to TCLASS DFHTCL00.

In addition, IBM supplies some other transaction classes. Transaction classes are defined using an eight-character name that can be used by the user to document the type of transaction class being defined such as CPUHOG to identify transactions that consume a lot of CPU resources. The TCLASS assignment is a two steps process: First step is to define a TCLASS (Figure 2) and the second step is to assign the transaction definition to the defined TCLASS (Figure 3). So, the major reason for using TCLASS to control input transactions is that through this mechanism the user can control specific types of transactions to be executed. It allows the user to limit the number of a specific type of transaction to enter the system.

There are two important definition figures when defining a TCLASS. The first identifies the maximum number of tasks (MAXACTIVE) that are going to be allowed into the system via a particular TCLASS before CICS/TS begins to queue the incoming transactions. MAXACTIVE limits the transactions within this TCLASS group to a specified maximum value. For example, if the MAXACTIVE is set to 10, then once there are ten transactions in this TCLASS in the system, the eleventh one and higher are queued for attachment. The TCLASS control occurs before the MXT control takes effect

```
OBJECT CHARACTERISTICS                                CICS RELEASE = 0640
CEDA View TRANClass( RESTRICT )
TRANClass      : RESTRICT
Group          : CWSDAC
Description    : DEMO TCLASS ←
CLASS LIMITS
Maxactive      : 001           0-999
Purgethresh   : 0000004      No | 1-1000000

SYSID=C310 APPLID=CICSTS31

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
04B █                               06 2                               01/03
```

Figure 2. Transaction Class RDO Definition

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0640
CEDA Alter TRANsAction( TREK )
+ DYNamic      ==> No           No | Yes
ROutable      ==> No           No | Yes
REMOTESystem  ==>
REMOTENAME    ==>
TRProf        ==>
Localq        ==>           No | Yes
SCHEDULING
PRIOrity      ==> 255          0-255
TClass        : No           No | 1-10
TRANClass     ==> DFHTCLOO ←
ALIASES
ALias         ==>
TASKReq       ==>
XTRanid       ==>
TPName        ==>
+ XTPname     :

SYSID=C310 APPLID=CICSTS31

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
04B █                               07 4                               04/22
```

Figure 3. Defining a Transaction to a Particular TCLASS

In addition, TCLASS allows the user to limit the number of transactions queued for entry into a particular TCLASS through the use of the parameter PURGETHRESH. This parameter is used to limit the number of transactions queuing for eligibility into a particular TCLASS. So, using our previous example of a MAXACTIVE of 10, if we specify a PURGETHRESH of 5, then we would allow a queue of up to four transactions with the fifth and higher being purged automatically by CICS/TS. The four queued transactions would be eligible for attach when one of the original TCLASS members completes. This feature allows the user to control the amount of storage being used by queuing transactions. It should be noted that queued transactions do not occupy a lot of storage because they have not completed the entire attach process yet.

Like MXT, TCLASS is usually an inherited parameter that was widely used when the CPU systems lacked the horsepower and real storage that is available in today's modern processors. In addition, it was an effective control for transactions that ran below the line. However, processors have become more powerful today and have more real storage available. In addition, many applications were migrated over the line during the Y2K conversion period. There are generally four major reasons for using TCLASS, which are:

- 1) To control resource hungry transactions. These are transactions that can use a lot of CPU cycles, require a lot of virtual or real storage and/or do a lot of I/O (e.g., heavy browsing) in the system. Limiting the number of these types of transactions that can execute at any time is a way to ensure that other transactions are not affected by these resource "hogs".
- 2) Single threading to control or protect resources. This may not be the best way to achieve this protection but it works and is simple to implement without major programming changes.
- 3) To control the number of transactions executing below the line. This is a method of ensuring that you don't run out of virtual storage below the line while at the same time being able to maintain a high MXT that will not affect the performance of transactions running above the line.
- 4) To avoid MRO/ISC "sympathy sickness". This allows the user to be able to control how many tasks can enter the system that have to go to another system by setting the MAXACTIVE combined with the PURGETHRESH to a figure that is being used for the inter-system communication sessions.

There may be other reasons but these have been the most prevalent at customer sites.

The use of TCLASS is not free. The system has to access the TCLASS information when the transaction is attached through a lookup of the TCLASS table by the Directory Domain. At transaction attach, the system can determine if there is a membership slot available within the TCLASS and must update the counters. The table entry has to be located at task termination to

once again update the counters. This adds CPU processing to the transaction execution. If TCLASS is no longer needed then this extra CPU overhead is wasted resources.

The first thing to analyze is the peak number of transactions executed versus the MAXACTIVE specified. If the peak is less than 50% of the MAXACTIVE, you should analyze why you are using TCLASS to control this class of transactions. This condition may occur because the amounts of CPU resources (cycles and real storage) available today are proportionally higher than they were ten years ago. If the TCLASS is needed for some reason (like those previously mentioned), then maybe the MAXACTIVE may require adjustment to a lower value. The opposite is also true. Maybe you are reaching the MAXACTIVE value and delaying transactions for TCLASS reasons. Then you must review if there are sufficient resources available that were not used because the transactions were waiting due to TCLASS. As mentioned previously, unused resources are gone and cannot be saved for a rainy day. It is a bad thing to hold back transactions when there is no paging, no virtual storage constraint and the CPU is running at 50%. This is an example of an “artificial delay” that adversely affects response times. In this case, you should either increase the MAXACTIVE value to allow more transactions to execute or eliminate the control. This is an installation dependant decision.

The correct setting of MXT and use of the TCLASS parameters in CICS/TS can have a positive effect on performance. Both can be very useful and require monitoring to ensure that MXT and TCLASS are working within the boundaries that were defined. Incorrect use can result in wasted resources and/or create an “artificial bottleneck”.

If you have any questions, comments or request for more information, please contact us:

Address: C\TREK Corporation PO BOX 560069 MONTVERDE FL 34756	Phone: (407) 469-3600 C\TREK Corporation (787) 756-5620 Advanced Computer Technology
E-mail: ctrek@actpr.com	Fax: (787) 756-5150
Website: www.ctrekcorp.com	Support: (787) 397-4150 (321) 297-5838 (787) 462-0406