#SHAREorg

SHARE
Technology · Connections · Results

CICS Common Performance Problems and Debugging

Ed Addison
IBM

February 6 2013
12440

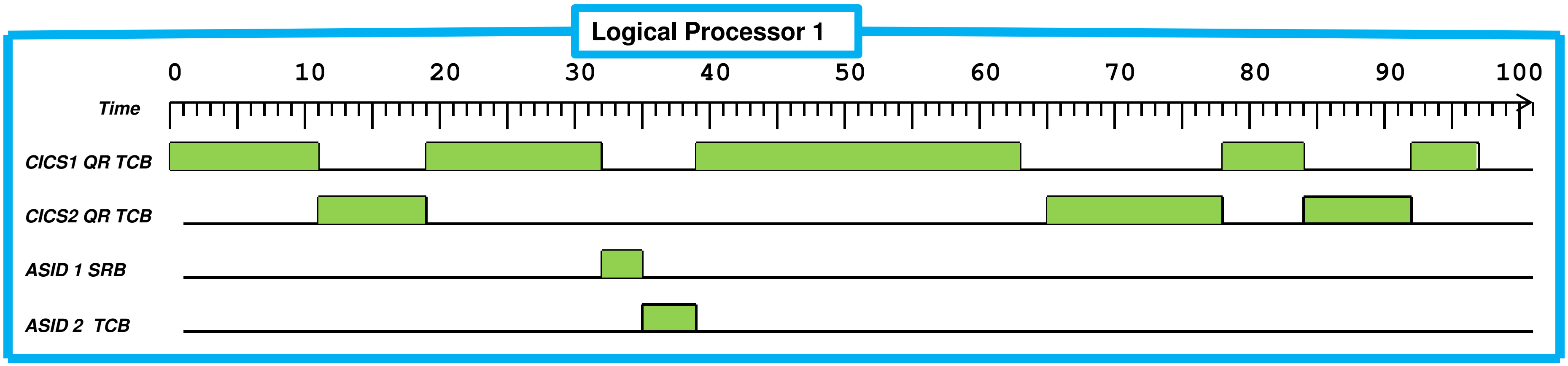
SHARE in Anaheim 2012

Agenda

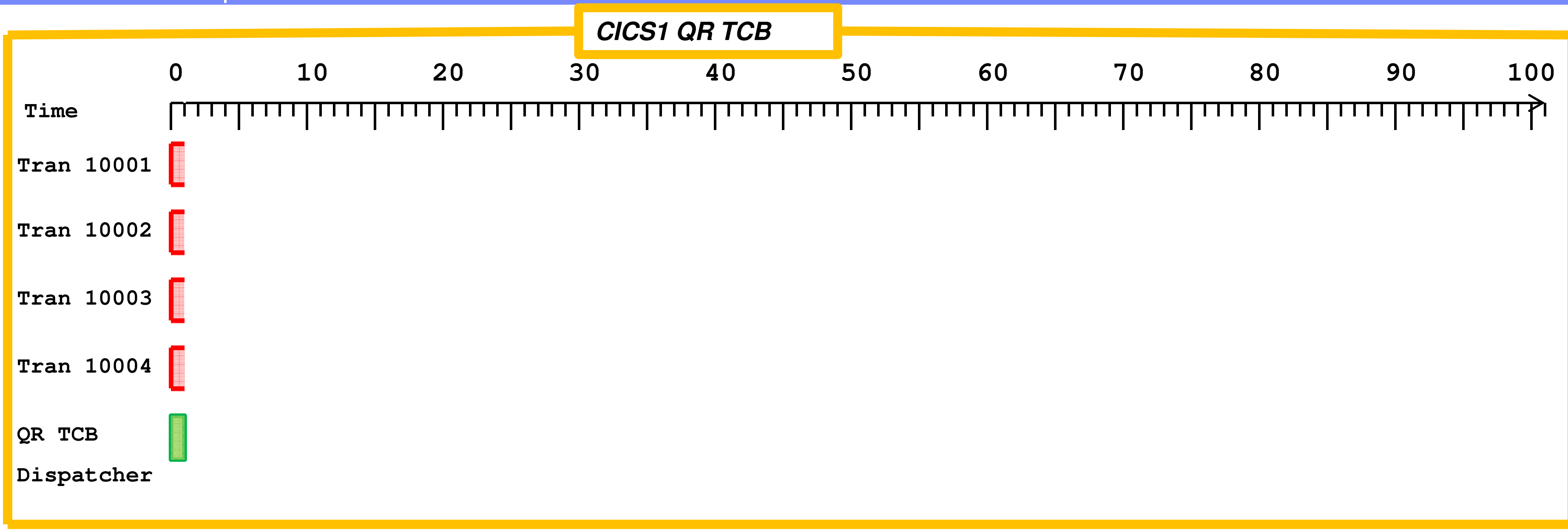
- CICS Dispatcher Basics
- Performance Problem - Loop
- Externalize MXT with CICS System Events
- CICS Monitoring Facility
- RMFIII
- Systrace perfdata

CICS Dispatcher Basics

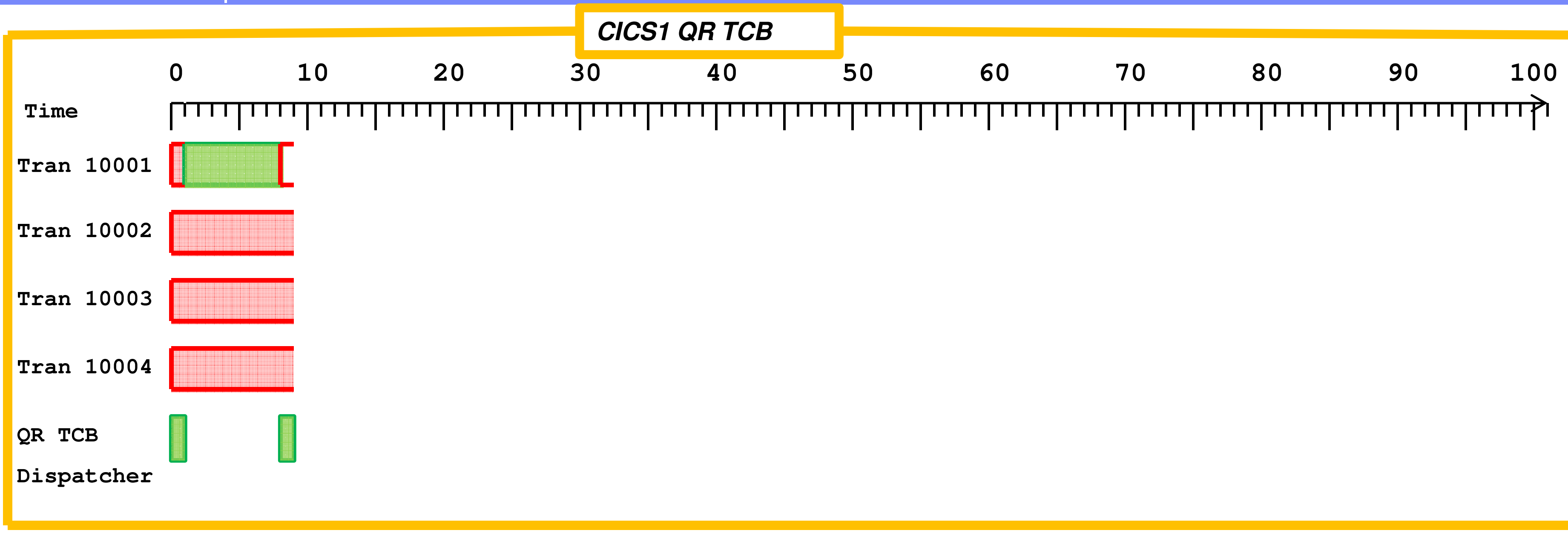
- **The box in blue below shows TCBs and SRBs using Logical Processor 1 in an LPAR.**
- **Only one thing (TCB or SRB) can run at a time on this processor.**
- **z/OS decides which TCBs and SRBs run on the processor.**
- **A typical well-behaved Concurrency(quasirent) CICS application program does not usually do anything that would cause the QR TCB to wait or suspend the QR TCB to the z/OS dispatcher.**
  - **But, there is nothing to stop this from occurring.**



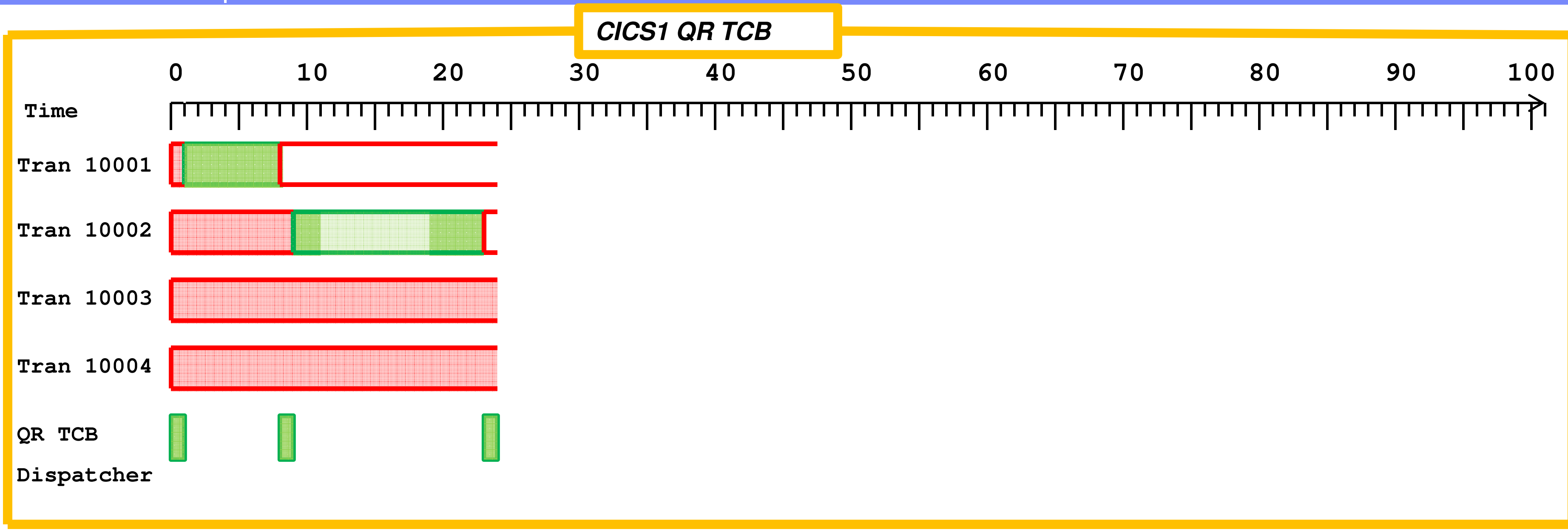
- There are several different ways that one TCB or SRB can lose or relinquish control of the processor.
  - **A TCB can be interrupted while it is executing instructions. Then z/OS can give control of the processor to a higher priority TCB or SRB. The interrupted TCB is left undispatched until z/OS gives it a processor and it can then resume executing instructions.**
  - **A TCB can voluntarily give up control by suspending or waiting to the z/OS dispatcher.**
    - **That can happen explicitly. For instance, when the CICS dispatcher has no CICS transactions ready to run on the TCB it will issue an SVC 1 wait to temporarily give control back to z/OS so something else can use the processor.**
  - **Paging is another way that a TCB can lose control. If an instructions needs a page of storage that has to be paged in from Aux, then the TCB gives up control of the processor and waits for the I/O. In the interim, the z/OS dispatcher can find another TCB or SRB who wants to run on the processor.**



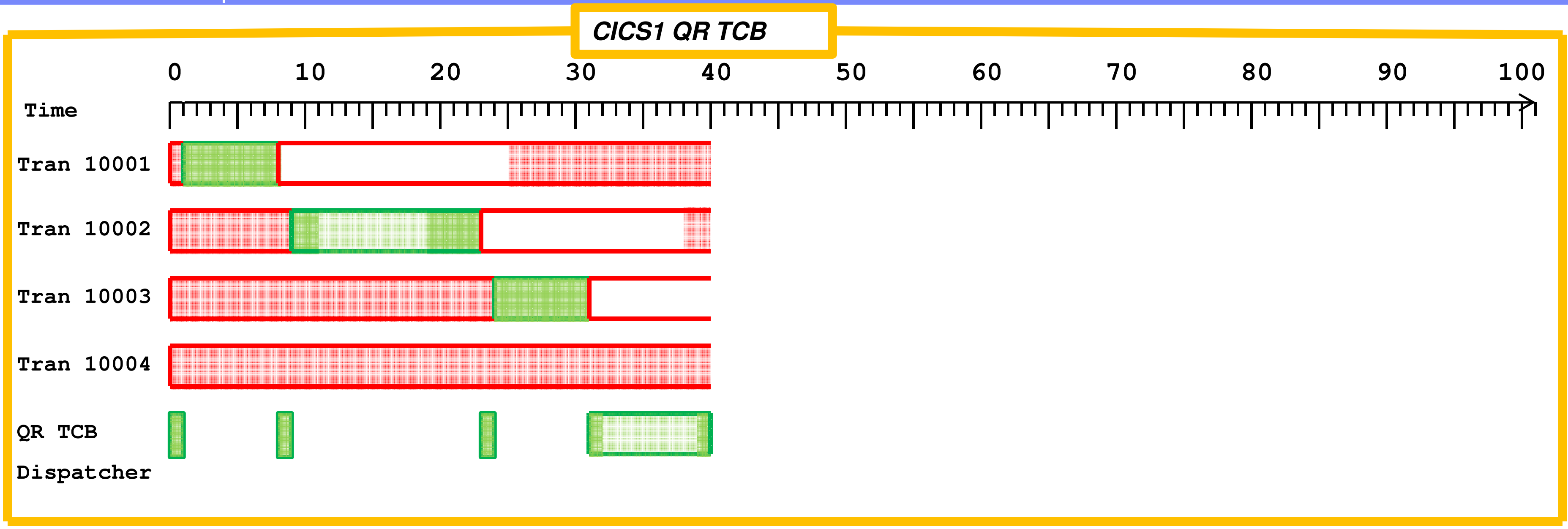
- On the QR TCB, CICS has built its own dispatching environment. In that environment CICS transactions share the QR TCB.
- The CICS Dispatcher decides which transactions run on the QR TCB.
- Here, 4 transactions have all just been attached and are all ready to run on the QR TCB. They are all Dispatchable as indicated by the light red shading.
- The CICS Dispatcher picks one and gives it control.



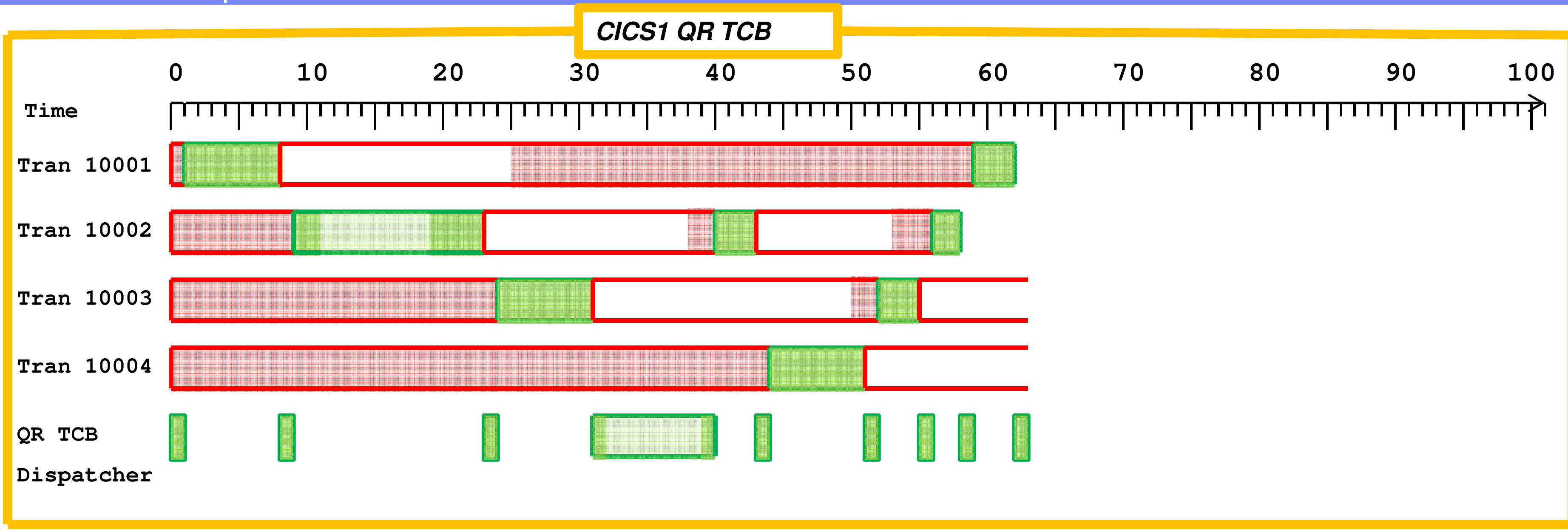
- Tran 10001 received control. It held control of the TCB for 7 units of wall-clock time and then it suspended giving control back to the CICS dispatcher.
- When the CICS dispatcher gets control, it knows how long transaction 10001 had control of the QR TCB, but it doesn't know how much CPU it used until it asks z/OS with a TIMEUSED. We'll say it used 7 units of CPU too. That is indicated by the solid green shading in the box.
- The dispatcher now chooses 1 of the 3 dispatchable tasks to dispatch.



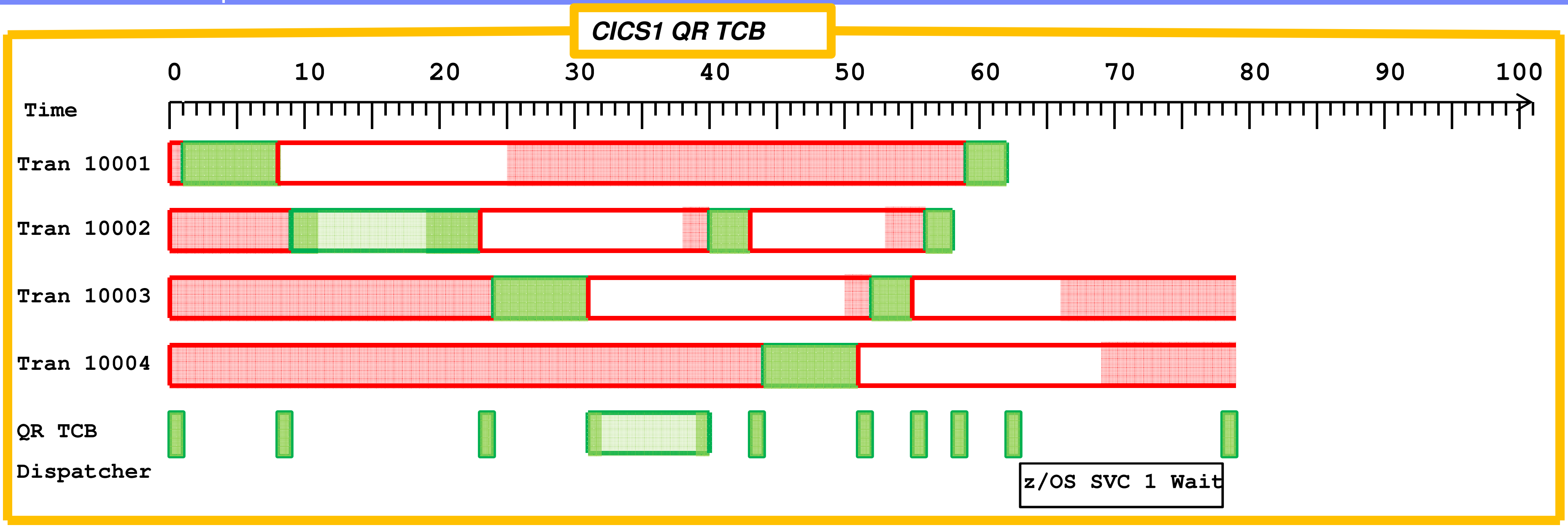
- Transaction 10002 had control of the TCB for 13 units of time. But this time it used only 7 units of CPU. So while transaction 10002 was in control of the QR TCB, the QR TCB stopped executing instructions for some reason. Maybe z/OS took control away to let higher priority work use the processor.
- Transaction 10001 is still suspended. Transaction 10002 just suspended. 10003 and 10004 are both dispatchable. The CICS dispatcher picks one of them.



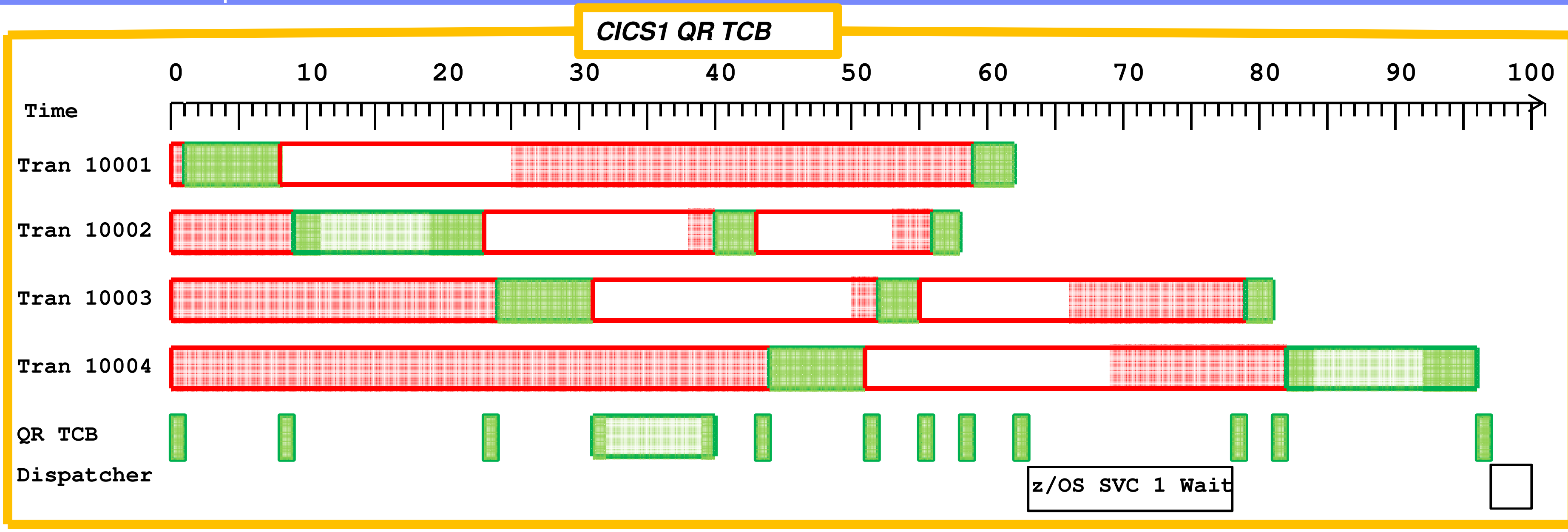
- Tran 10003 ran solidly and suspended.
- Then while in the CICS dispatcher code, control of the QR TCB was taken away for about 7 units of time.
- During that time, at timeline 38, transaction 10002 became dispatchable. That means that whatever it was suspended on has completed. An example is when suspended for File I/O. When the I/O completes and the ECB is posted, the waiting CICS transaction immediately becomes dispatchable.



- Here, several transactions ran, each time giving control back to the CICS dispatcher.
- Transactions 10001 and 10002 have finished.
- At this point, control is in the CICS dispatcher but there are no dispatchable tasks. When that happens, the CICS dispatcher issues an SVC 1 Wait to give control of the TCB back to z/OS temporarily.



- During that time, notice 2 transactions became dispatchable. Typically the TCB would wake up out of its wait immediately when a transaction becomes dispatchable. The reason that didn't happen is probably because the processor was not available.



- Here the last 2 tasks finish up. Transaction 10004 lost control of the TCB for some reason during its last dispatch.
- Since there are no more transactions, the CICS Dispatcher issues another SVC1 Wait.

Problem One: Loop

Problem One - Loop

- Customer called the Support Center for no transactions running yet high CPU consumed by the CICS region
- **ST SYS** (Status SYStem)
  - With a dump, from the IPCS Commands panel, enter the **ST SYS** command to find out what time the dump was taken. Below is an example of the output

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
  Sysplex name: EDZPLEX
  TIME OF DAY CLOCK: C2D443EE B58366C4  08/11/2008 18:31:27.786295 local
  TIME OF DAY CLOCK: C2D3D8A4 E38366C4  08/11/2008 23:31:27.786295 GMT
  Program Producing Dump: SVCDUMP
  Program Requesting Dump: IEAVTSDT
  Incident token: EDZPLEX 07/08/2008 23:31:27.198911 GMT
```

- When getting information from this screen, it is important to note both the LOCAL Time-Of-Day and the GMT Time-Of-Day
  - The CICS Dispatcher summary gives times as GMT rather than LOCAL

VERBX DFHPDxxx ‘CSA=2’

```
CSA=2

=== SUMMARY OF ACTIVE ADDRESS SPACES

     ASID(hex):          JOBNAME:
     0148                CICS01



===CSA: COMMON SYSTEM AREA AND OPTIONAL FEATURES LIST

CSA 0004EF98 Common System Area

0000  00000248 0004B020 101BA578 800BADE4  90669D14 0ECF6F30 116C8528 116C87FC *..........v....U......?..%e..%g.*
0020  9066DDD0 9066E70E 116C8C30 1021FEE0  131DE000 7F3FB960 0004E770 11AFC030 *...}..X..%.....\..\."..-..X...{.*
0040  00051020 00054080 0011400C 102C3680  1624268F 11D9D8C8 00000100 00000000 *......  ...  ..........RQH........*
0060  005A2101 00000000 00090D70 000038DA  00000000 00000000 7FFFFFFF 0106189F *.!....................."........*
0080  00AC6000 E0004800 00009080 8ECF0090  000033A3 00000000 001E001E E707E762 *..-.\.............t........X.X.*
```

- **In this example, the CSA time is 16:24:26.8 LOCAL Time**
- **There is other important information given in this output:**
  - **The jobname of CICS**
  - **The address space ID (ASID) of CICS**

Compare Times

- Once you have the CSA time and the LOCAL time from ST SYS, you can decide if CICS is healthy or not
  - If there is a several minute time gap between the two times, then you know CICS is unhealthy
  - If the two times are close, then you know CICS is healthy and something else is causing the problem
- Even when a healthy CICS is dumped, there is usually some difference between the two times. This is because ST SYS is not the exact time CICS started dumping
  - If there is less than a minute difference between CSA time and SY SYS LOCAL time, then you can generally say CICS was healthy when the dump was taken
- From this example, the CSA has not been updated in over 2 hours:
  - CSA time is: 16:24:26.8 local
  - ST SYS time is: 18:31:27.7 local
- If the difference between CSA time and ST SYS local time leads you to believe CICS is unhealthy, then this should coincide with CICS CPU utilization
  - When CICS is unhealthy, it is either getting no CPU time (hung) or it is getting all the CPU time (looping)

Is CICS looping or is it hung?

- In this example CICS is not healthy. This indicates the CICS Dispatcher is not getting control for one of 2 reasons:
  - The CICS Dispatcher has given control to a CICS task, and the CICS task has never given control back
  - The CICS Dispatcher has given up control to z/OS, and z/OS has never redispatched CICS
- To determine which one it is, enter **VERBX DFHPDxxx ‘KE=1’**

VERBX DFHPDxxx 'KE=1'

```
===KE: Kernel Domain KE_TASK Summary

KE_NUM KE_TASK   STATUS         TCA_ADDR TRAN_# TRANSID DS_TASK  KE_KTCB  ERROR

0001   0EC54C80 KTCB Step     00000000                 00000000 0EC96080
0002   0EC54900 KTCB QR       00000000                 10203030 0EC99020
0003   0EC54580 KTCB RO       00000000                 10203148 0EC98040
0004   0EC54200 KTCB CO       00000000                 10203260 1012B020
0005   0EC71C80 KTCB FO       00000000                 10203378 0EC97060
0006   0EC71900 Not Running   00000000                 10136080 0EC98040
0007   0EC71580 Unused
0008   0EC71200 KTCB SL       00000000                 102035A8 10169020
0009   0EC8EC80 Not Running   00000000                 101F3680 0EC99020
000A   1026E400 KTCB CQ       00000000                 10203490 10146020
...
0024   101EB900 ***Running**  00000000                 10136380 10146020
...
01A4   116C6080 ***Running**  102C3680 84551  CSPG     101CD580 0EC99020
```

VERBX DFHPDxxx 'KE=1'(cont)

- Look to see if there is a ***Running** task under the QR TCB. If there is, then the CICS Dispatcher has given control to the task, and the task has not given control back
  - You first need to find the address of the QR KTCB. It is in the KE_KTCB column on the line showing the KTCB QR in the STATUS column
  - In the previous example, you can see there is a running task dispatched on the QR TCB
  - Note: There is another running task, dispatched on the CQ TCB. This is the console/KILL task which remains available for console requests or requests to KILL a looping or hung task
- The task on the QR TCB is 'running' from the CICS Dispatcher's perspective. This simply means it has never given control back to the CICS Dispatcher
  - It could be looping
  - It could have done something causing the CICS QR TCB to lose control
- Find out by using the z/OS System trace and the CICS trace
- Before you look at the z/OS System trace, you need to know the ASID of CICS, and the TCB address of the QR TCB

Find the address of the QR TCB

- Find the address of the QR TCB by listing the contents of the QR KTCB
  - **IP L 0EC99020 ASID(x'148') L(999)**
    - **We obtained this address on slide 18**
  - Offset x'50' into a KTCB is the address of the corresponding z/OS TCB:

```
LIST 0EC99020. ASID(X'0148') LENGTH(X'03E7') AREA
ASID(X'0148') ADDRESS(0EC99020.) KEY(80)
0EC99020. D2E3C3C2 40404040 00000000 0EC54900  116C6080 0EC5C020 00000159 6D263B20 |KTCB    .....E..E{....._...|
0EC99040. 00000000 7D000000 00000000 00000000  80000004 00000000 36800000 D800D8D9 |....'.................Q.QR|
0EC99060. 00000000 00000000 0EC99020 40000000  00AEB5D8 00000000 00006120 00000000 |.........I.. ......../.....|
0EC99080. 00000000 00000000 00000000 00000000  00000000 00000520 0000038E 00000008 |..........................|
0EC990A0. 00AEB5D8 00000000 1012B020 0EC98040  00000000 00000000 BF12AAB3 4FC4FF88 |...Q.........I. .......|D.h|
```

- **An alternate way to identify the QR TCB is to format the CICS trace table.**

Enter VERBX DFHPDxxx 'TR=2', then do a FIND on QR

```
AP 4D01 CQCQ  EXIT  - FUNCTION(MERGE_CIB_QUEUES) RESPONSE(OK)

        TASK-TCP   KE_NUM-001C TCB-QR    /00AEB5D8 RET-905E1C0A TIME-16:20:56.9458823764
```

- **Now see what the z/OS System trace indicates**

z/OS System Trace Table

- **From IPCS Option 6 Command enter:**
  - **SYSTRACE TIME(LOCAL)**

```
SYSTRACE TIME(LOCAL)
-------------------------------------- SYSTEM TRACE TABLE -----------------------
--                                     ------------------
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  ... PASD SASD TIMESTAMP-LOCAL  CP
06 0148 00AEB5D8  EXT   1005 078D0000 929888E2  ... 0148 0148 18:31:25.338251  01
06 0148 00AEB5D8  I/O  00458 078D2000 929888E6  ... 0148 0148 18:31:25.338279  01
06 0148 00AEB5D8  EXT   1005 078D0000 929888EE  ... 0148 0148 18:31:25.338675  01
06 0148 00AEB5D8  I/O  034D4 078D2000 929888EA  ... 0148 0148 18:31:25.338763  01
06 0148 00AEB5D8  EXT   1005 078D0000 929888EE  ... 0148 0148 18:31:25.339097  01
06 0148 00AEB5D8  I/O  03DF2 078D0000 929888E2  ... 0148 0148 18:31:25.339261  01
06 0148 00AEB5D8  EXT   1005 078D0000 929888EE  ... 0148 0148 18:31:25.339519  01
06 0148 00AEB5D8  CLKC       078D0000 929888E2  ... 0148 0148 18:31:25.339861  01
06 0148 00AEB5D8  DSP        078D0000 929888E2  ... 0148 0148 18:31:25.340108  01
06 0148 00AEB5D8  I/O  0045A 078D2000 929888E6  ... 0148 0148 18:31:25.340167  01
06 0148 00AEB5D8  I/O  00458 078D2000 929888EA  ... 0148 0148 18:31:25.340264  01
06 0148 00AEB5D8  EXT   1005 078D0000 929888E2  ... 0148 0148 18:31:25.340535  01
06 0148 00AEB5D8  I/O  03DF2 078D0000 929888E2  ... 0148 0148 18:31:25.340787  01
06 0148 00AEB5D8  EXT   1005 078D2000 929888EA  ... 0148 0148 18:31:25.340957  01
06 0148 00AEB5D8  I/O  03DF2 078D2000 929888EA  ... 0148 0148 18:31:25.341303  01
06 0148 00AEB5D8  EXT   1005 078D0000 929888EE  ... 0148 0148 18:31:25.341380  01
03 0148 00AEB5D8  DSP        078D0000 929888EE  ... 0148 0148 18:31:25.341420  04
```

- Note: 00AEB5D8 is the QR TCB derived from the previous slide

System Trace Table (cont)

- Verify the ASID being traced is the one for the CICS region we care about
  - If it isn't, you can enter **SYSTRACE TIME(LOCAL) ASID(x'xx')**
- The TCB address shows up in the second column. Verify it is the QR TCB
- This trace shows a loop on the QR TCB. Notice the PSW address on the DSP and EXT trace entries
- Before we saw this trace, we already knew CICS was unhealthy
  - A CICS task had not relinquished control to the CICS Dispatcher for over two hours
- By looking at the System trace table, we can verify if the CICS task was looping, or if it had lost control to z/OS
  - Since we see trace entries for the QR TCB, we assume it is looping

CICS is Looping

- What you expect to see in the System trace table is a looping pattern. In this example, we have a pattern of DSP and EXT trace entries
  - EXT trace entries are an external interrupt
    - z/OS is taking control away from the TCB in order to process some sort of interrupt (an I/O interrupt in this case)
  - DSP trace entries are z/OS Dispatcher trace entries
    - The z/OS Dispatcher is giving control back to the TCB at the exact instruction address where control was taken
  - I/O trace entries are z/OS high priority interrupts when I/O finishes
  - CLKC trace entries are z/OS checking clocks when z/OS services haven't been requested for awhile
- By looking at the PSW addresses in the System trace, you can begin to learn what module(s) comprise the loop
  - If there are several modules involved in the loop, you would likely need to look at lots of I/O, EXT, DSP entries before you could get a handle on the extent of the loop
  - In this example, it is clear fairly quickly the problem is a tight loop involving only a few instructions between address 129888E2 and 129888EE

Finding the Looping Program

- The next step is to identify the program(s) in which the looping instructions live
- If you are in SYSTRACE, and want to know what module a PSW address falls within, you first need to subtract the high-order bit (the x'80' bit, if there is one)
  - For instance, if the PSW address is 81234568, then the address you need to use is 1234568
  - If the PSW address is A1234568, then the address you need to use will be 21234568
- Once you have the address aaaaaaaa, you have several choices for figuring out the module:
  - **VERBX DFHPDxxx 'LD=1'** displays the Loader Domain summary information
    - Enter **FIND 'PROGRAM STORAGE MAP'**
    - The Program Storage Map lists the modules loaded by CICS, in address order
    - In our example, for PSW address 929888E2, we could do a FIND on ' 129' to get closer to the programs listed near this address

VERBX DFHPDxxx ‘LD=1’

- **VERBX DFHPDxxx ‘LD=1’**

```
==LD: PROGRAM STORAGE MAP

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

 DFHCSA   8004E200 DFHKELCL 0004D000 650   HCI6700  06/05/11 05.51     1        1
                   -noheda- 0004D4F8
                   DFHKELRT 0004D500 650   HCI6700  06/05/11 05.51
                   -noheda- 0004D8F8
                   DFHCSAOF 0004D900 0650 HCI6700  I   05/11 06.53
                   DFHCSA   0004E000 0650 HCI6700  I   05/11 06.53
                   DFHKERCD 0004E4B0 650   HCI6700  06/05/11 05.51

... then FIND on `  129' shows:

DFHCRS   92982D70 DFHCRS   12982D50 0650          I   29/11 23.23     1        0
DFHSNP   92984BE8 DFHYA630 12984BC0 630                               1        0
                  DFHSNP   12984C58 0650          I   30/11 02.48
DFHTPR   92987FFA DFHTPR   12987FD0 0650 HCI6700  I   13/12 11.09     1        1
GGGGPEND 9298D538 DFHYA640 1298D510 640                               1        0
```

Using the WHERE command or Browse mode

- Enter **WHERE aaaaaaaa** or simply **W aaaaaaaa** from a command line
  - If the first digit of the address starts with a letter, then you could enter the WHERE command followed by a period:
    - **WHERE aaaaaaa.** e.g. **WHERE C00498.**
  - Or you could include a leading zero:
    - **WHERE 0aaaaaaa** e.g. **WHERE 0C00498**
- The WHERE command is useful when CICS doesn't know about the module
  - WHERE is not helpful in this example because the looping module was loaded by CICS, not z/OS, so z/OS is unable to identify the module. **WHERE 129888E2** displays:

```
ASID(X'0148') 129888E2. AREA(Subpool252Key00)+1888E2 IN EXTENDED PRIVATE
```

- You can also try to display the PSW address in Browse mode and back up looking for an eyecatcher

The CICS Trace Table

- If the System trace entries indicate the loop is larger than a tight loop within one module, it is possible CICS services are being requested by the looping module. If this is true and if CICS internal trace is active, then you may be able to see the loop in the CICS trace

  Note: Some CICS services (like EXEC CICS SUSPEND or EXEC CICS SEND WAIT) cause the task to be suspended and the CSA Time-of-Day clock to be updated. Because we have determined CICS is unhealthy in this discussion, we know no such services are being requested. Other CICS services (like EXEC CICS ASSIGN or EXEC CICS FREEMAIN) do not cause the task to be suspended (i.e. do not give control back to the CICS Dispatcher)

- To format the internal CICS trace table, enter **VERBX DFHPDxxx 'TR=2'**

CICS Trace

- **VERBX DFHPDxxx ‘TR=2’**

- The trace entries below are the last trace entries in the dump. The time stamps match the CSA Time-of-Day. This is consistent with a tight loop. As soon as the tight loop starts, there are no more CICS trace entries, no more updates of the CSA Time-of-Day, and no more useful work done by CICS

```
AP 1940 APLI  ENTRY - FUNCTION(START_PROGRAM) PROGRAM(DFHTPR) CEDF_STATUS(NOCEDF) EXECUTION_SET(FULLAPI)
                   SYNCONRETURN(NO) LANGUAGE_BLOCK(1174EAE0) COMMAREA(00000000 , 00000000) LINK_LEVEL(1)

              TASK-84551 KE_NUM-01A4 TCB-QR   /00AEB5D8 RET-9047C94E TIME-16:24:26.8881611423

XM 1001 XMIQ  ENTRY - FUNCTION(INQUIRE_TRANSACTION)
              TASK-84551 KE_NUM-01A4 TCB-QR   /00AEB5D8 RET-9059A64C TIME-16:24:26.8881644079

XM 1002 XMIQ  EXIT  - FUNCTION(INQUIRE_TRANSACTION) RESPONSE(OK) FACILITY_TYPE(TERMINAL) TRANNUM(0084551C)
                   ORIGINAL_TRANSACTION_ID(CSPG)

              TASK-84551 KE_NUM-01A4 TCB-QR   /00AEB5D8 RET-9059A64C TIME-16:24:26.8881673220

---TRACE TABLE END---
```

```
NOTE:  Time to call the Support Center for loop in DFH module
```

Externalize CICS MaxTask with System Event

Externalize MXT with System Events

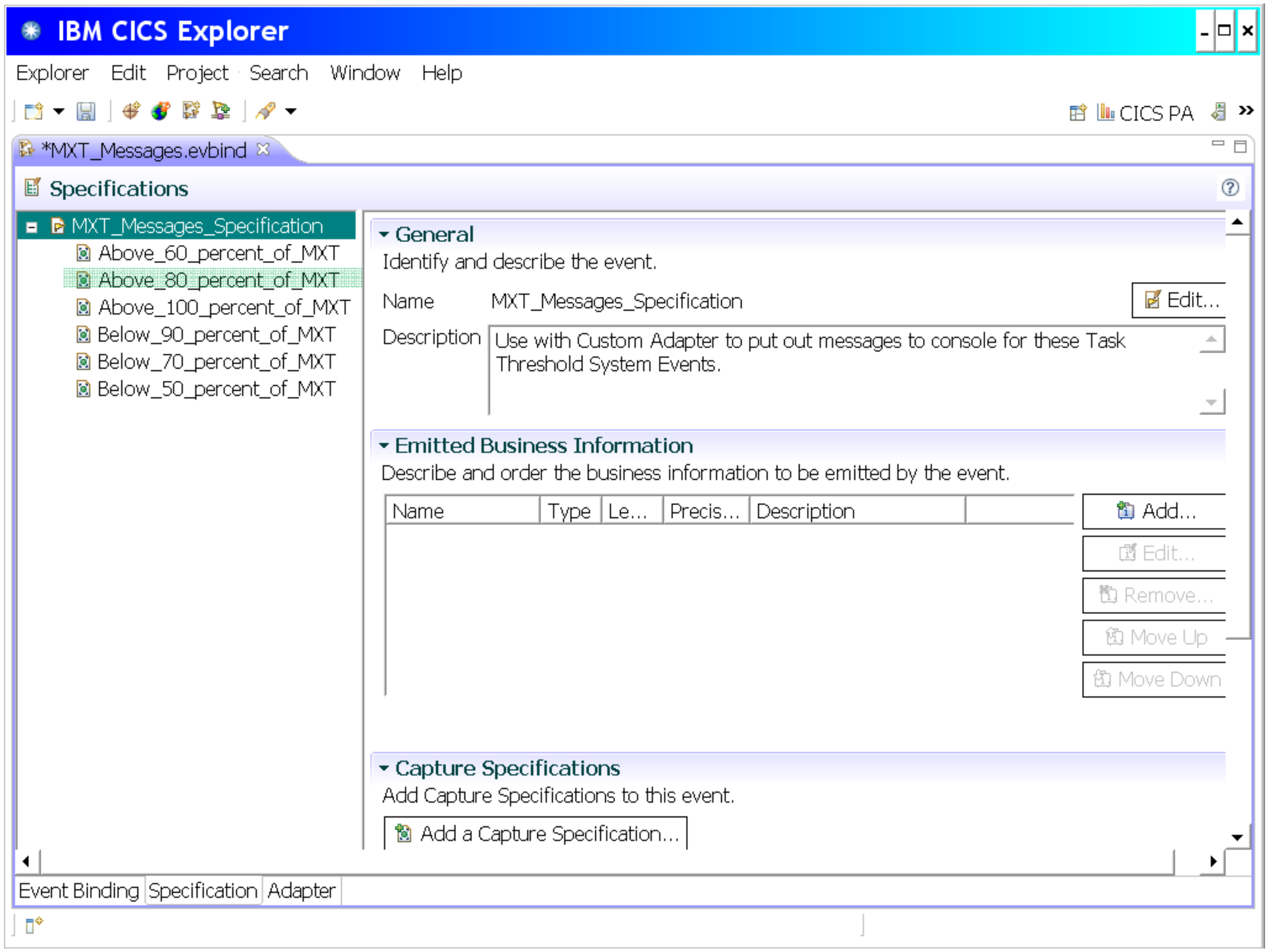
- New with CICS Transaction Server 4.2
- Event processing supports the following system events:
  - FILE enable or disable status
  - FILE open or close status
  - DB2CONN connection status
  - TASK threshold
  - TRANCLASS TASK threshold
  - Unhandled transaction abend
- Use CICS Explorer to build Event Binding file
- Prepare and install a Transaction and a Program that will write out a console message at various task thresholds.
- Set a SLIP to get a dump on one of the messages.

Problem: IYNXK went MaXTask

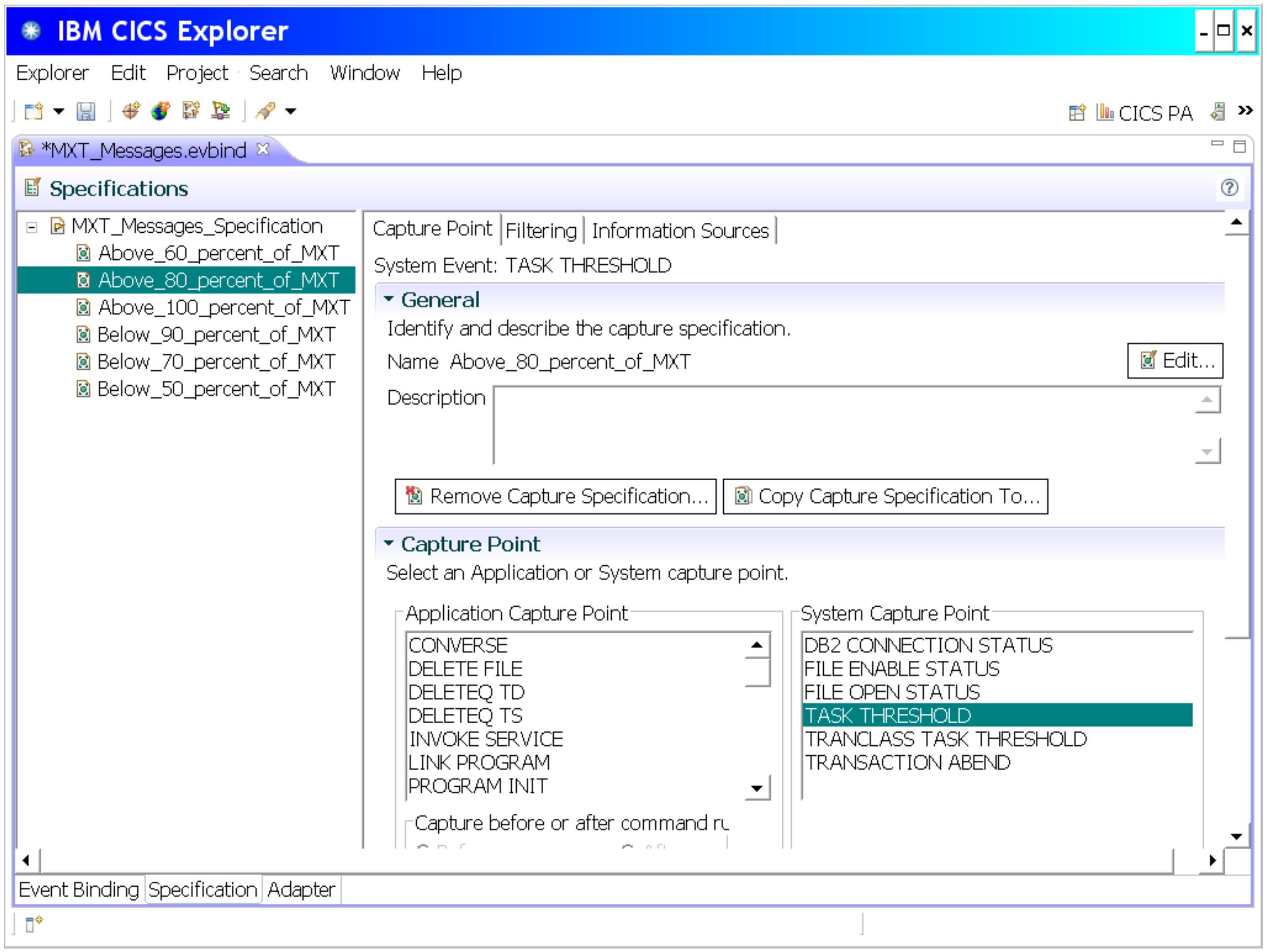
```
07.10.08 JOB18137  +ABOVE_60_PERCENT_OF_MXT
07.10.14 JOB18137  +ABOVE_80_PERCENT_OF_MXT
07.10.17 JOB18137  IEA794I SVC DUMP HAS CAPTURED:  032
   032             DUMPID=154 REQUESTED BY JOB (IYNXK   )
   032             DUMP TITLE=SLIP DUMP ID=AB80
07.10.20 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.10.24 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.10.26 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.10.36 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.10.38 JOB18137  +BELOW_70_PERCENT_OF_MXT
07.10.45 JOB18137  +ABOVE_80_PERCENT_OF_MXT
07.10.51 JOB18137  +BELOW_70_PERCENT_OF_MXT
07.10.53 JOB18137  +BELOW_50_PERCENT_OF_MXT
07.11.00 JOB18137  IEA794I SVC DUMP HAS CAPTURED:  073
   073             DUMPID=155 REQUESTED BY JOB (*MASTER*)
   073             DUMP TITLE=IYNXK MXT
07.11.18 JOB18137  +ABOVE_60_PERCENT_OF_MXT
07.11.20 JOB18137  +ABOVE_80_PERCENT_OF_MXT
07.11.22 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.11.28 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.11.30 JOB18137  +BELOW_70_PERCENT_OF_MXT
```

```
07.11.32 JOB18137  +ABOVE_80_PERCENT_OF_MXT
07.11.34 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.11.46 JOB18137  IEA794I SVC DUMP HAS CAPTURED:  117
   117             DUMPID=156 REQUESTED BY JOB (*MASTER*)
   117             DUMP TITLE=IYNXK MXT2
07.12.00 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.12.01 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.12.06 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.12.07 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.12.12 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.12.12 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.12.17 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.12.18 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.12.26 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.12.28 JOB18137  +ABOVE_100_PERCENT_OF_MXT
07.13.08 JOB18137  +BELOW_90_PERCENT_OF_MXT
07.13.10 JOB18137  +BELOW_70_PERCENT_OF_MXT
07.13.11 JOB18137  +BELOW_50_PERCENT_OF_MXT
07.15.27 JOB18137  IEA794I SVC DUMP HAS CAPTURED:  197
   197             DUMPID=157 REQUESTED BY JOB (*MASTER*)
   197             DUMP TITLE=IYNXK NORM
```

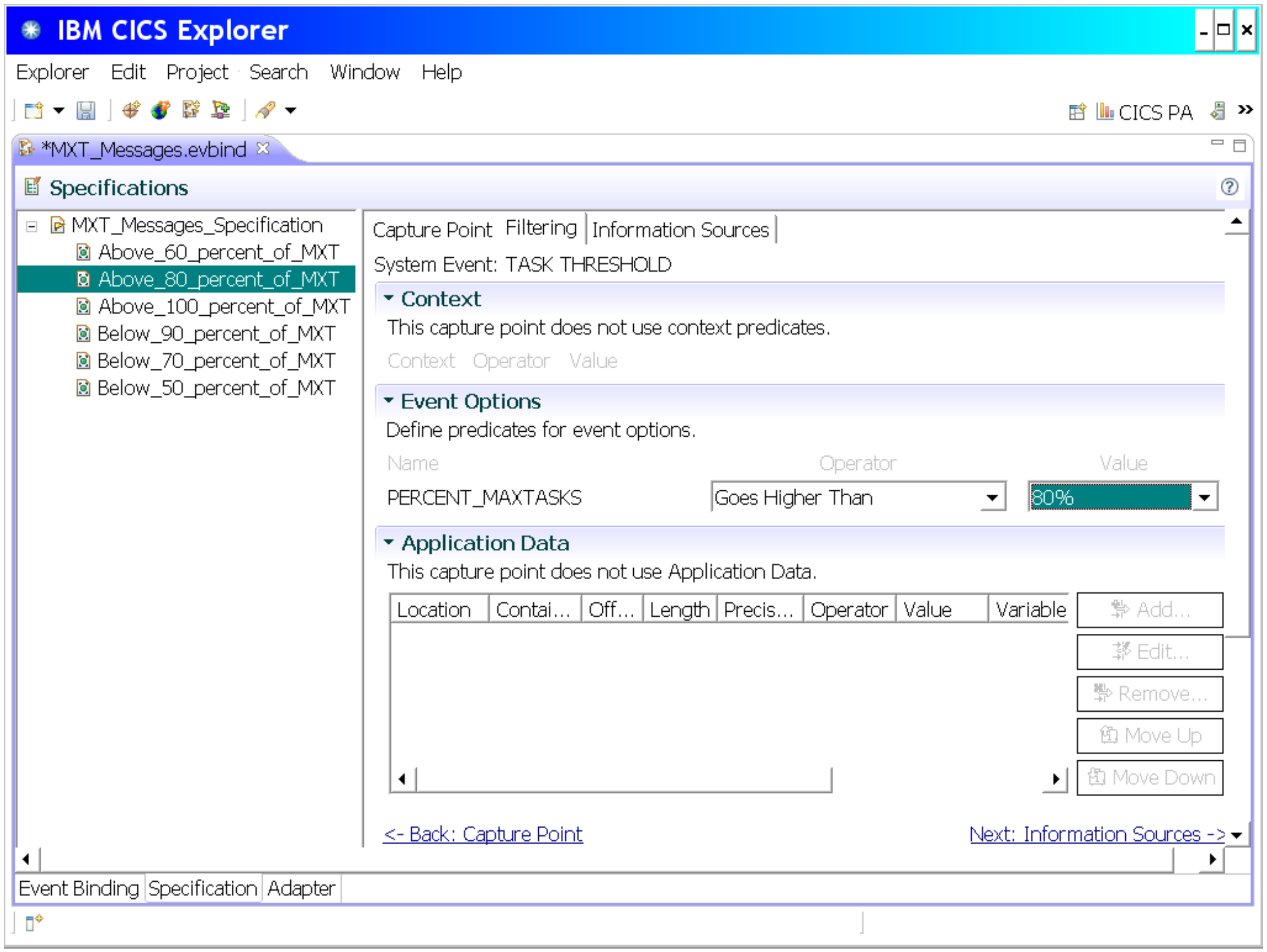
Create an Event Binding Specification that contains 6 Capture Specifications as shown. The name of each Capture Specification is the content of the message sent to the console.



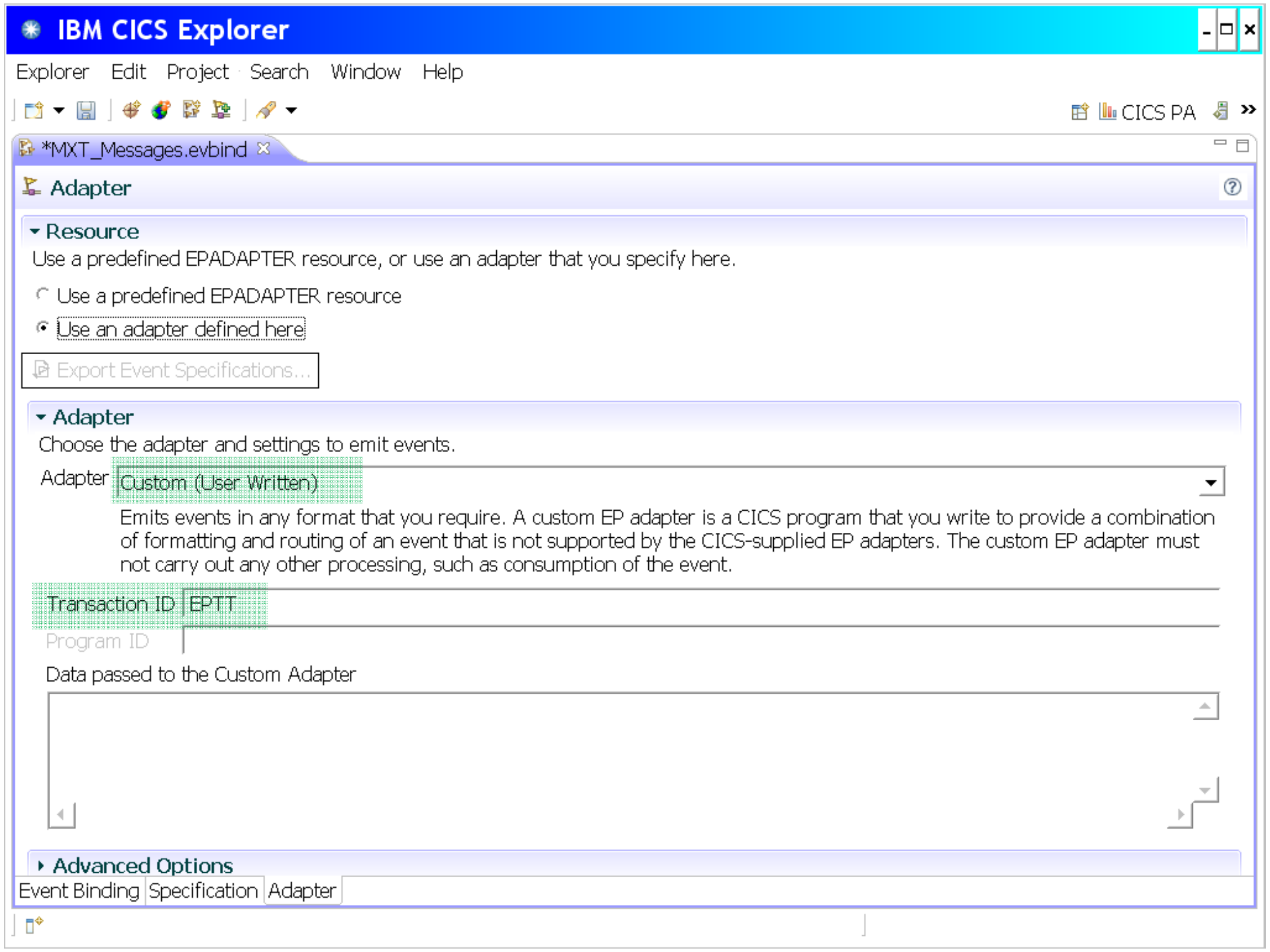
For each Capture Specification, choose a TASK THRESHOLD System Capture Point



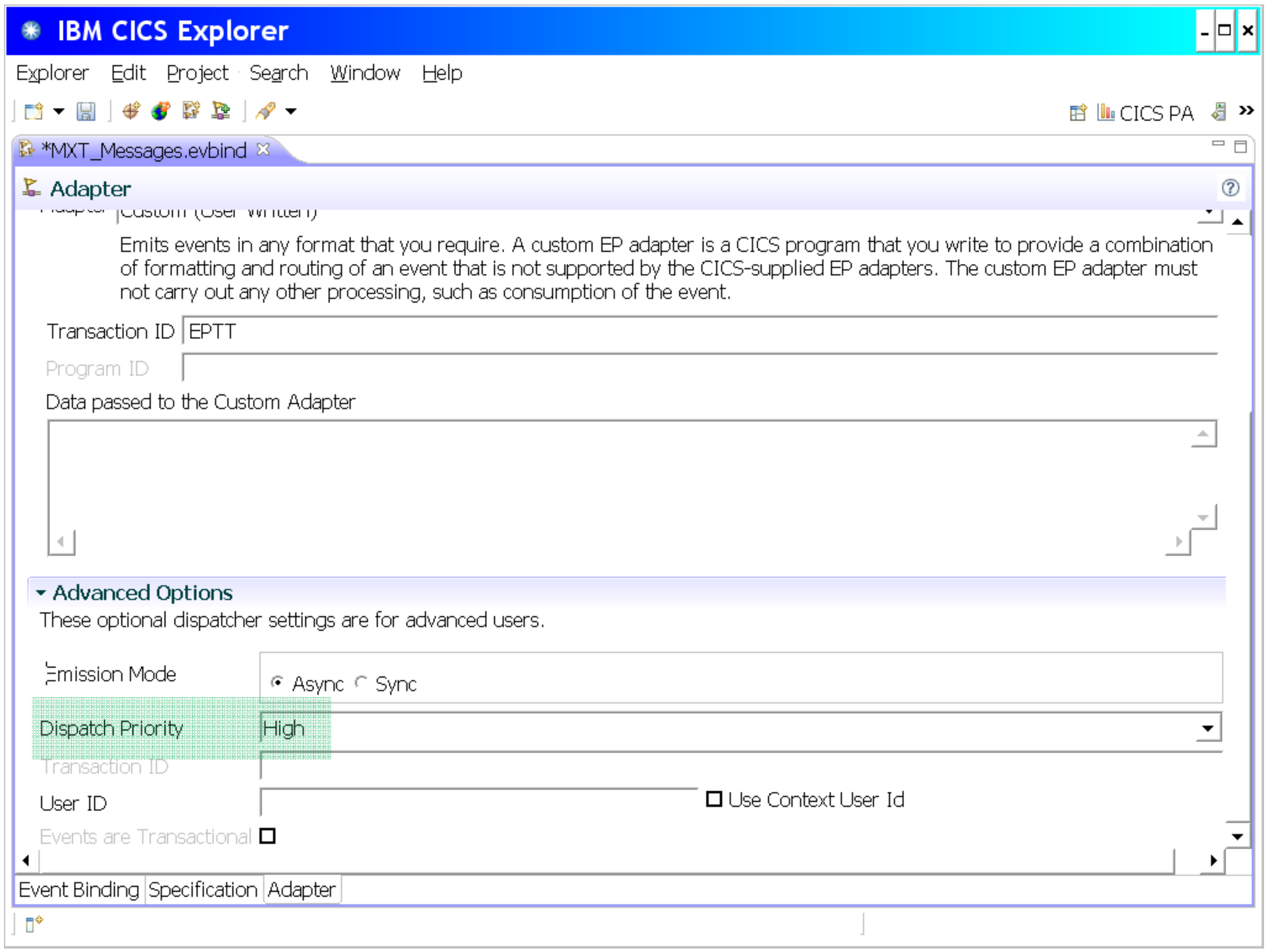
For each Capture Specification, define a predicate that matches the name of the Capture Specification.



For the Adapter, choose Custom (User Written) and put in a Transaction ID. Then click on Advanced Options.



In advanced Options, let everything default except specify Dispatch Priority High.



Translate, Assemble, and Link the following program into a dataset in the DFHRPL concatenation

```
 TITLE 'EPADAPTR'
***********************************************************************
* EPADAPTR: Puts out a message to the console                        *
*---------------------------------------------------------------------*
DFHEISTG DSECT
STRUCLEN DS    CL4
*
         DFHREGS
         COPY  DFHEPCXD      Covers DFHEP.CONTEXT container
         COPY  DFHEPDED      Covers DFHEP.DESCRIPTOR container
         COPY  DFHEPAPD      Covers DFHEP.ADAPTPARM container
*
EPADAPTR CSECT
EPADAPTR AMODE ANY
EPADAPTR RMODE ANY
*
         EXEC CICS GET CONTAINER('DFHEP.CONTEXT')                      X
                   SET(R9) FLENGTH(STRUCLEN)
         USING EPCX,R9
         EXEC CICS WRITE OPERATOR TEXT(EPCX_CS_NAME)
         EXEC CICS RETURN
*
         END
```

Final Steps:

- Export the Bundle Project containing the Event Binding Specification.
- Define and Install a Transaction definition for EPTT and a Program definition for EPADAPTR. Specify Priority(255) on the EPTT transaction definition.
- Using the exported Bundle Project file, define and install the Bundle
- And if you want to get a dump on one of the messages, here is a SLIP:

```
SLIP SET,MSGID='ABOVE_80',J=jobname,ID=AB80,A=SVCD,ML=1,END
```

CICS Monitoring Facility Information

- Two CICS/PA summary forms
- Use them with the 4 example tasks
- Use them with the problem SMF110 data

SUSPSUM summarizes components of Suspend Time.

DISPSUM summarizes components of Dispatch Time.

```
SUMMARY(OUTPUT(SUSPSUM),
        EXTERNAL(CPAXW001),
        TOTALS(8),
        INTERVAL(00:00:30),
        FIELDS(START(TIMES,ASCEND),
               TASKCNT,
               RESPONSE(AVE),
               DISPATCH(TIME(AVE)),
               SUSPEND(TIME(AVE)),
               SUSPEND(COUNT(AVE)),
               DSPDELAY(TIME(AVE)),
               MXTDELAY(TIME(AVE)),
               TCLDELAY(TIME(AVE)),
               DISPWAIT(TIME(AVE)),
               QRMODDLY(TIME(AVE)),
               FCWAIT(TIME(AVE)),
               FCWAIT(COUNT(AVE)))),
```

```
SUMMARY(OUTPUT(DISPSUM),
        EXTERNAL(CPAXW002),
        TOTALS(8),
        INTERVAL(00:00:30),
        FIELDS(START(TIMES,ASCEND),
               TASKCNT,
               RESPONSE(AVE),
               SUSPEND(TIME(AVE)),
               DISPATCH(TIME(AVE)),
               CPU(TIME(AVE)),
               QRDISPT(TIME(TOT)),
               QRDISPT(TIME(AVE)),
               QRCPU(TIME(TOT)),
               KY8DISPT(TIME(AVE)),
               KY8DISPT(COUNT(AVE)),
               L8CPU(TIME(AVE)),
               MXTDELAY(TIME(AVE))))
```

SUSPSUM Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0
07:09:00	3276	.1562	.0091	.1471	1	.0830	.0000	.0000	.0638	.0638	.0641	0
07:09:30	3228	.3328	.0093	.3234	1	.1698	.0000	.0000	.1525	.1524	.1528	0
07:10:00	2285	2.1023	.0137	2.0886	1	1.0377	.0289	.0000	1.0375	1.0375	1.0076	0
07:10:30	2105	1.5692	.0131	1.5561	1	.7964	.0083	.0000	.7540	.7540	.7083	0
07:11:00	2384	1.1418	.0125	1.1293	1	.5423	.0434	.0000	.5813	.5813	.5195	0
07:11:30	1945	3.4445	.0158	3.4287	1	1.8043	.3032	.0000	1.6064	1.6064	1.4462	0
07:12:00	2446	2.4340	.0117	2.4223	1	1.2851	.1436	.0000	1.1246	1.1246	.9916	0
07:12:30	3240	1.7993	.0091	1.7902	1	.9038	.0030	.0000	.8778	.8778	.8823	0
07:13:00	3051	.6163	.0091	.6072	1	.3217	.0000	.0000	.2806	.2806	.2843	0
07:13:30	3252	.0753	.0091	.0661	1	.0413	.0000	.0000	.0246	.0246	.0248	0
07:14:00	3258	.1391	.0091	.1300	1	.0742	.0000	.0000	.0556	.0556	.0559	0
07:14:30	3258	.1640	.0091	.1548	1	.0867	.0000	.0000	.0679	.0679	.0682	0

- Average Response Time started going bad in the 7:09:30 interval.
- It was back to normal starting in the 7:13:30 interval.
- You can see from the MXTDelay column which intervals had some MXT delay.

SUSPSUM Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0
07:09:00	3276	.1562	.0091	.1471	1	.0830	.0000	.0000	.0638	.0638	.0641	0
07:09:30	3228	.3328	.0093	.3234	1	.1698	.0000	.0000	.1525	.1524	.1528	0
07:10:00	2285	2.1023	.0137	2.0886	1	1.0377	.0289	.0000	1.0375	1.0375	1.0076	0
07:10:30	2105	1.5692	.0131	1.5561	1	.7964	.0083	.0000	.7540	.7540	.7083	0
07:11:00	2384	1.1418	.0125	1.1293	1	.5423	.0434	.0000	.5813	.5813	.5195	0
07:11:30	1945	3.4445	.0158	3.4287	1	1.8043	.3032	.0000	1.6064	1.6064	1.4462	0
07:12:00	2446	2.4340	.0117	2.4223	1	1.2851	.1436	.0000	1.1246	1.1246	.9916	0
07:12:30	3240	1.7993	.0091	1.7902	1	.9038	.0030	.0000	.8778	.8778	.8823	0
07:13:00	3051	.6163	.0091	.6072	1	.3217	.0000	.0000	.2806	.2806	.2843	0
07:13:30	3252	.0753	.0091	.0661	1	.0413	.0000	.0000	.0246	.0246	.0248	0
07:14:00	3258	.1391	.0091	.1300	1	.0742	.0000	.0000	.0556	.0556	.0559	0
07:14:30	3258	.1640	.0091	.1548	1	.0867	.0000	.0000	.0679	.0679	.0682	0

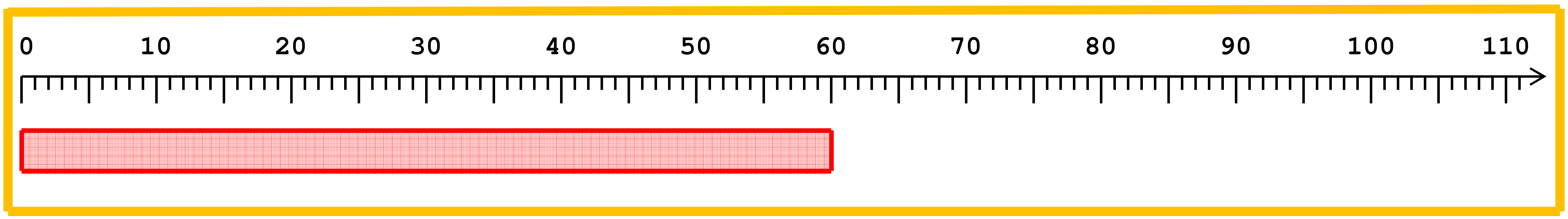
- Notice that Response time is always Dispatch time plus Suspend time. A task is always either Suspended or Dispatched.

SUSPSUM Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0
07:09:00	3276	.1562	.0091	.1471	1	.0830	.0000	.0000	.0638	.0638	.0641	0
07:09:30	3228	.3328	.0093	.3234	1	.1698	.0000	.0000	.1525	.1524	.1528	0
07:10:00	2285	2.1023	.0137	2.0886	1	1.0377	.0289	.0000	1.0375	1.0375	1.0076	0
07:10:30	2105	1.5692	.0131	1.5561	1	.7964	.0083	.0000	.7540	.7540	.7083	0
07:11:00	2384	1.1418	.0125	1.1293	1	.5423	.0434	.0000	.5813	.5813	.5195	0
07:11:30	1945	3.4445	.0158	3.4287	1	1.8043	.3032	.0000	1.6064	1.6064	1.4462	0
07:12:00	2446	2.4340	.0117	2.4223	1	1.2851	.1436	.0000	1.1246	1.1246	.9916	0
07:12:30	3240	1.7993	.0091	1.7902	1	.9038	.0030	.0000	.8778	.8778	.8823	0
07:13:00	3051	.6163	.0091	.6072	1	.3217	.0000	.0000	.2806	.2806	.2843	0
07:13:30	3252	.0753	.0091	.0661	1	.0413	.0000	.0000	.0246	.0246	.0248	0
07:14:00	3258	.1391	.0091	.1300	1	.0742	.0000	.0000	.0556	.0556	.0559	0
07:14:30	3258	.1640	.0091	.1548	1	.0867	.0000	.0000	.0679	.0679	.0682	0

- Get used to what is normal. Dispatch time is normally about .0091. That increases significantly during the problem intervals. Suspend time is normally about .1300. That increases significantly during the problem intervals.

SUSPSUM Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0

DISPSUM Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000



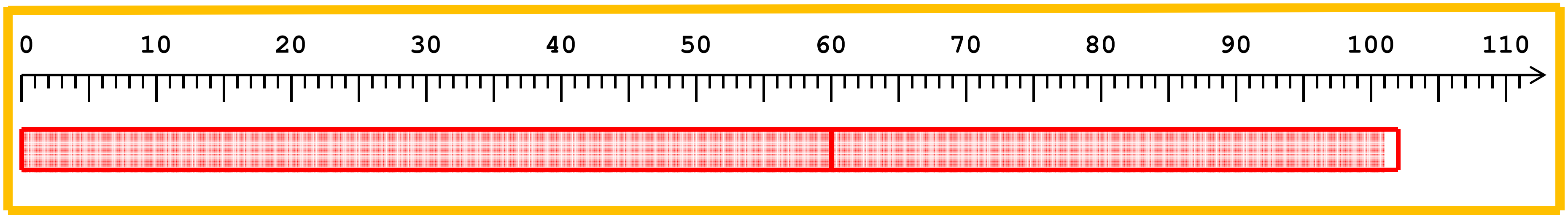
- Let's graph the 07:08:30 30-second interval. It is a normal, pre-problem interval.
- Disp1Dly is 60 milliseconds and there is no MXTDelay or TCLDelay. So all 60 milliseconds is dispatchable, waiting to run on the QR.

SUSPSUM

Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0

DISPSUM

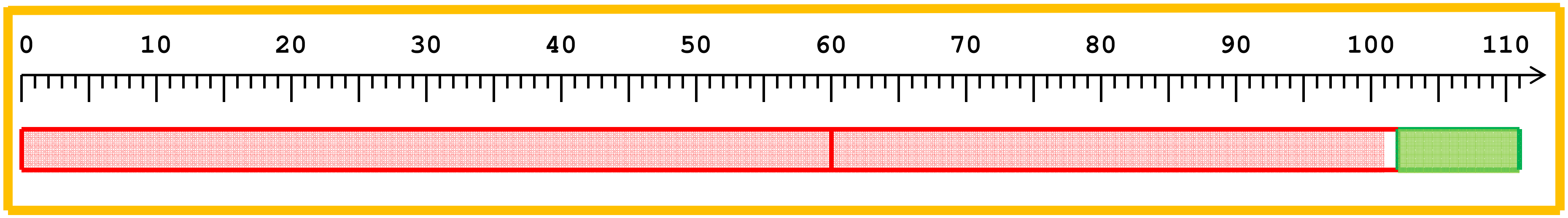
Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000



- Suspend Time is 102 milliseconds and Disp1Dly is 60 milliseconds. So the remaining part of Suspend time is 42 milliseconds. Of that, 41 milliseconds is waiting for redispatch (DispWait) on the QR (QRModDly).
- So, almost the whole 102 millisecond suspend time is waiting to run on the QR. Clearly the QR TCB is a bottleneck, during normal intervals.

SUSPSUM		Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg	Avg
Start	#Tasks	Response	Dispatch	Suspend	Suspend	Disp1Dly	MXTDelay	TCLDelay	DispWait	QRModDly	FC Wait	FC Wait
Interval		Time	Time	Time	Count	Time	Time	Time	Time	Time	Time	Count
07:08:30	3228	.1113	.0092	.1021	1	.0605	.0000	.0000	.0414	.0414	.0416	0

DISPSUM		Avg	Avg	Avg	Avg	Total	Avg	Avg	Avg	Avg	Avg	Avg
Start	#Tasks	Response	Suspend	Dispatch	User CPU	QR Disp	QR Disp	QR CPU	KY8 Disp	KY8 Disp	L8 CPU	MXTDelay
Interval		Time	Time	Time	Time	Time	Time	Time	Time	Count	Time	Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000



- Dispatch Time is 9 milliseconds. Notice that QR Disp is the same. So we know that the transactions only ran on the QR TCB.
- User CPU (and QR CPU) round up to 9 milliseconds. So we'll make the whole 9 milliseconds dark green.

DISPSUM Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000
07:09:00	3276	.1562	.1471	.0091	.0086	29.6774	.0091	.0086	.0000	0	.0000	.0000
07:09:30	3228	.3328	.3234	.0093	.0088	30.1325	.0093	.0088	.0000	0	.0000	.0000
07:10:00	2285	2.1023	2.0886	.0137	.0115	31.3524	.0137	.0115	.0000	0	.0000	.0289
07:10:30	2105	1.5692	1.5561	.0131	.0115	27.4879	.0131	.0115	.0000	0	.0000	.0083
07:11:00	2384	1.1418	1.1293	.0125	.0115	29.8614	.0125	.0115	.0000	0	.0000	.0434
07:11:30	1945	3.4445	3.4287	.0158	.0117	30.8260	.0158	.0117	.0000	0	.0000	.3032
07:12:00	2446	2.4340	2.4223	.0117	.0106	28.6731	.0117	.0106	.0000	0	.0000	.1436
07:12:30	3240	1.7993	1.7902	.0091	.0086	29.5015	.0091	.0086	.0000	0	.0000	.0030
07:13:00	3051	.6163	.6072	.0091	.0086	27.8617	.0091	.0086	.0000	0	.0000	.0000
07:13:30	3252	.0753	.0661	.0091	.0086	29.7519	.0091	.0086	.0000	0	.0000	.0000
07:14:00	3258	.1391	.1300	.0091	.0086	29.6127	.0091	.0086	.0000	0	.0000	.0000
07:14:30	3258	.1640	.1548	.0091	.0086	29.6975	.0091	.0086	.0000	0	.0000	.0000

- Here is the DISPSUM form showing dispatch time fields.
- Notice that Dispatch Time and QR Disp Time are the same. That means that all processing is on the QR TCB.

DISPSUM Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000
07:09:00	3276	.1562	.1471	.0091	.0086	29.6774	.0091	.0086	.0000	0	.0000	.0000
07:09:30	3228	.3328	.3234	.0093	.0088	30.1325	.0093	.0088	.0000	0	.0000	.0000
07:10:00	2285	2.1023	2.0886	.0137	.0115	31.3524	.0137	.0115	.0000	0	.0000	.0289
07:10:30	2105	1.5692	1.5561	.0131	.0115	27.4879	.0131	.0115	.0000	0	.0000	.0083
07:11:00	2384	1.1418	1.1293	.0125	.0115	29.8614	.0125	.0115	.0000	0	.0000	.0434
07:11:30	1945	3.4445	3.4287	.0158	.0117	30.8260	.0158	.0117	.0000	0	.0000	.3032
07:12:00	2446	2.4340	2.4223	.0117	.0106	28.6731	.0117	.0106	.0000	0	.0000	.1436
07:12:30	3240	1.7993	1.7902	.0091	.0086	29.5015	.0091	.0086	.0000	0	.0000	.0030
07:13:00	3051	.6163	.6072	.0091	.0086	27.8617	.0091	.0086	.0000	0	.0000	.0000
07:13:30	3252	.0753	.0661	.0091	.0086	29.7519	.0091	.0086	.0000	0	.0000	.0000
07:14:00	3258	.1391	.1300	.0091	.0086	29.6127	.0091	.0086	.0000	0	.0000	.0000
07:14:30	3258	.1640	.1548	.0091	.0086	29.6975	.0091	.0086	.0000	0	.0000	.0000

- This chart summarizes all the tasks that started during each 30-second interval. Notice how, even during the good intervals, Total QR Disp time is very close to 30 seconds. This is further evidence that the QR TCB is a bottleneck. That squares with how almost all of the Suspend time is waiting to run on the QR TCB.

DISPSUM Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000
07:09:00	3276	.1562	.1471	.0091	.0086	29.6774	.0091	.0086	.0000	0	.0000	.0000
07:09:30	3228	.3328	.3234	.0093	.0088	30.1325	.0093	.0088	.0000	0	.0000	.0000
07:10:00	2285	2.1023	2.0886	.0137	.0115	31.3524	.0137	.0115	.0000	0	.0000	.0289
07:10:30	2105	1.5692	1.5561	.0131	.0115	27.4879	.0131	.0115	.0000	0	.0000	.0083
07:11:00	2384	1.1418	1.1293	.0125	.0115	29.8614	.0125	.0115	.0000	0	.0000	.0434
07:11:30	1945	3.4445	3.4287	.0158	.0117	30.8260	.0158	.0117	.0000	0	.0000	.3032
07:12:00	2446	2.4340	2.4223	.0117	.0106	28.6731	.0117	.0106	.0000	0	.0000	.1436
07:12:30	3240	1.7993	1.7902	.0091	.0086	29.5015	.0091	.0086	.0000	0	.0000	.0030
07:13:00	3051	.6163	.6072	.0091	.0086	27.8617	.0091	.0086	.0000	0	.0000	.0000
07:13:30	3252	.0753	.0661	.0091	.0086	29.7519	.0091	.0086	.0000	0	.0000	.0000
07:14:00	3258	.1391	.1300	.0091	.0086	29.6127	.0091	.0086	.0000	0	.0000	.0000
07:14:30	3258	.1640	.1548	.0091	.0086	29.6975	.0091	.0086	.0000	0	.0000	.0000

- How is it possible for tasks that ran in a 30-second interval to use more than 30 seconds of QR Disp time? It is because these intervals include all tasks that started within the interval. For example, tasks that started at 07:10:29.9 are included in the 07:10:00 interval even though all of their processing is after 07:10:30.

DISPSUM		Avg	Avg	Avg	Avg	Total	Avg	Avg	Avg	Avg	Avg	Avg
Start	#Tasks	Response	Suspend	Dispatch	User CPU	QR Disp	QR Disp	QR CPU	KY8 Disp	KY8 Disp	L8 CPU	MXTDelay
Interval		Time	Time	Time	Time	Time	Time	Time	Time	Count	Time	Time
07:08:30	3228	.1113	.1021	.0092	.0086	29.6445	.0092	.0086	.0000	0	.0000	.0000
07:09:00	3276	.1562	.1471	.0091	.0086	29.6774	.0091	.0086	.0000	0	.0000	.0000
07:09:30	3228	.3328	.3234	.0093	.0088	30.1325	.0093	.0088	.0000	0	.0000	.0000
07:10:00	2285	2.1023	2.0886	.0137	.0115	31.3524	.0137	.0115	.0000	0	.0000	.0289
07:10:30	2105	1.5692	1.5561	.0131	.0115	27.4879	.0131	.0115	.0000	0	.0000	.0083
07:11:00	2384	1.1418	1.1293	.0125	.0115	29.8614	.0125	.0115	.0000	0	.0000	.0434
07:11:30	1945	3.4445	3.4287	.0158	.0117	30.8260	.0158	.0117	.0000	0	.0000	.3032
07:12:00	2446	2.4340	2.4223	.0117	.0106	28.6731	.0117	.0106	.0000	0	.0000	.1436
07:12:30	3240	1.7993	1.7902	.0091	.0086	29.5015	.0091	.0086	.0000	0	.0000	.0030
07:13:00	3051	.6163	.6072	.0091	.0086	27.8617	.0091	.0086	.0000	0	.0000	.0000
07:13:30	3252	.0753	.0661	.0091	.0086	29.7519	.0091	.0086	.0000	0	.0000	.0000
07:14:00	3258	.1391	.1300	.0091	.0086	29.6127	.0091	.0086	.0000	0	.0000	.0000
07:14:30	3258	.1640	.1548	.0091	.0086	29.6975	.0091	.0086	.0000	0	.0000	.0000

- Notice how the QR CPU time and the QR Disp time both suddenly increase. Given that the suspend time is almost all waiting for dispatch on the QR, it is clear that this sudden increase in QR Disp time has something to do with causing the MXT.

Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:09:55	108	.4217	.0093	.4124	1	.2163	.0000	.0000	.1959	.1959	.1961	0
07:09:56	108	.4078	.0090	.3988	1	.2099	.0000	.0000	.1887	.1887	.1889	0
07:09:57	108	.4226	.0091	.4136	1	.2106	.0000	.0000	.2027	.2027	.2029	0
07:09:58	94	.5417	.0121	.5296	1	.2645	.0000	.0000	.2650	.2650	.2652	0
07:09:59	80	.6383	.0130	.6253	1	.3111	.0000	.0000	.3140	.3140	.3142	0
07:10:00	88	.7077	.0124	.6954	1	.3449	.0000	.0000	.3502	.3502	.3504	0

Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:09:55	108	.4217	.4124	.0093	.0086	1.0027	.0093	.0086	.0000	0	.0000	.0000
07:09:56	108	.4078	.3988	.0090	.0086	.9758	.0090	.0086	.0000	0	.0000	.0000
07:09:57	108	.4226	.4136	.0091	.0086	.9801	.0091	.0086	.0000	0	.0000	.0000
07:09:58	94	.5417	.5296	.0121	.0115	1.1383	.0121	.0115	.0000	0	.0000	.0000
07:09:59	80	.6383	.6253	.0130	.0115	1.0429	.0130	.0115	.0000	0	.0000	.0000
07:10:00	88	.7077	.6954	.0124	.0116	1.0879	.0124	.0116	.0000	0	.0000	.0000

- Here we have the transactions summarized on 1-second intervals.
- With this we see that the point where the QR Disp and QR CPU times suddenly increased is actually at 07:09:58.

Start Interval	#Tasks	Avg Response Time	Avg Dispatch Time	Avg Suspend Time	Avg Suspend Count	Avg Disp1Dly Time	Avg MXTDelay Time	Avg TCLDelay Time	Avg DispWait Time	Avg QRModDly Time	Avg FC Wait Time	Avg FC Wait Count
07:09:55	108	.4217	.0093	.4124	1	.2163	.0000	.0000	.1959	.1959	.1961	0
07:09:56	108	.4078	.0090	.3988	1	.2099	.0000	.0000	.1887	.1887	.1889	0
07:09:57	108	.4226	.0091	.4136	1	.2106	.0000	.0000	.2027	.2027	.2029	0
07:09:58	94	.5417	.0121	.5296	1	.2645	.0000	.0000	.2650	.2650	.2652	0
07:09:59	80	.6383	.0130	.6253	1	.3111	.0000	.0000	.3140	.3140	.3142	0
07:10:00	88	.7077	.0124	.6954	1	.3449	.0000	.0000	.3502	.3502	.3504	0

Start Interval	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Avg QR CPU Time	Avg KY8 Disp Time	Avg KY8 Disp Count	Avg L8 CPU Time	Avg MXTDelay Time
07:09:55	108	.4217	.4124	.0093	.0086	1.0027	.0093	.0086	.0000	0	.0000	.0000
07:09:56	108	.4078	.3988	.0090	.0086	.9758	.0090	.0086	.0000	0	.0000	.0000
07:09:57	108	.4226	.4136	.0091	.0086	.9801	.0091	.0086	.0000	0	.0000	.0000
07:09:58	94	.5417	.5296	.0121	.0115	1.1383	.0121	.0115	.0000	0	.0000	.0000
07:09:59	80	.6383	.6253	.0130	.0115	1.0429	.0130	.0115	.0000	0	.0000	.0000
07:10:00	88	.7077	.6954	.0124	.0116	1.0879	.0124	.0116	.0000	0	.0000	.0000

- Prior to 07:09:58, there was a balance between transaction arrival rate and QR Disp time. Just enough transactions were arriving to keep the QR TCB totally busy. The 33% increase in QR Disp per task breaks that balance. Now the transactions are arriving faster than they can get their QR TCB time. So they back up.

RMFIII

- The problem is caused by transactions in IYNXK suddenly starting to use significantly more CPU at 7:09:58.
- Maybe RMFIII will yield some clues to help explain why that happened.

```
.                                    RMF Monitor III Primary Menu                 z/OS V1R12 RMF .
.  Selection ===> 2                                                                               .
.                                                                                                 .
.  Enter selection number or command on selection line.                                           .
.                                                                                                 .
.                                                                                                 .
.    S SYSPLEX           Sysplex reports and Data Index                              (SP) .
.    1 OVERVIEW          WFEX, SYSINFO, and Detail reports                           (OV) .
.    2 JOBS              All information about job delays                            (JS) .
.    3 RESOURCE          Processor, Device, Enqueue, and Storage                     (RS) .
.    4 SUBS              Subsystem information for HSM, JES, and XCF                (SUB) .
.                                                                                         .
.    U USER              User-written reports (add your own ...)                     (US) .
.                                                                                         .
.                                                                                         .
.                        O OPTIONS     T TUTORIAL     X EXIT                              .
.                                                                                         .
.        5694-A01 Copyright IBM Corp. 1986, 2010. All Rights Reserved                     .
.                     Licensed Materials - Property of IBM                                .
.                                                                                         .
.                                                                                         .
.                                                                                         .
.                                                                                         .
.                                                                                         .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE           .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE         .
```

- Type ‘2’ for Selection and press ENTER.

```
.                        RMF Job Report Selection Menu                          .
.  Selection ===> 5                                                             .
.                                                                               .
.  Enter selection number or command and jobname for desired job report.       .
.                                                                               .
.    Jobname ===> IYNXK___                                                      .
.                                                                               .
.    1 DEVJ             Delay caused by devices                     (DVJ)       .
.   1A DSNJ             .. Data set level                           (DSJ)       .
.    2 ENQJ             Delay caused by ENQ                          (EJ)       .
.    3 HSMJ             Delay caused by HSM                          (HJ)       .
.    4 JESJ             Delay caused by JES                          (JJ)       .
.    5 JOB              Delay caused by primary reason           (DELAYJ)       .
.    6 MNTJ             Delay caused by volume mount                (MTJ)       .
.    7 MSGJ             Delay caused by operator reply              (MSJ)       .
.    8 PROCJ            Delay caused by processor                    (PJ)       .
.    9 QSCJ             Delay caused by QUIESCE via RESET command    (QJ)       .
.   10 STORJ            Delay caused by storage                      (SJ)       .
.   11 XCFJ             Delay caused by XCF                          (XJ)       .
.                                                                               .
.  These reports can also be selected by placing the cursor on the              .
.  corresponding delay reason column of the DELAY or JOB reports and            .
.  pressing ENTER or by using the commands from any panel.                      .
.                                                                               .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE  .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE.
```

- Type ‘5’ for Selection and ‘IYNXK’ for Jobname and press ENTER.

```
                           RMF V1R12   Job Delays                          Line 1 of 3
  Command ===>                                                           Scroll ===> CSR

  Samples: 100     System: MV23  Date: 05/28/12  Time: 07.08.20  Range: 100    Sec

  Job: IYNXK         Primary delay: Job is waiting to use the processor.

  Probable causes: 1) Higher priority work is using the system.
                   2) Improperly tuned dispatching priorities.


  ------------------------- Jobs Holding the Processor --------------------------
  Job:           IYNXJ         Job:         RH23MSTR         Job:         OMEGTEMS
  Holding:          4%         Holding:           2%         Holding:           1%
  PROC Using:       9%         PROC Using:        2%         PROC Using:        1%
  DEV Using:        0%         DEV Using:         0%         DEV Using:         0%
  --------------------------- Job Performance Summary ---------------------------
          Service       WFL -Using%- DLY IDL UKN ---- % Delayed for ---- Primary
  CX ASID Class    P Cr  %   PRC DEV  %   %   %  PRC DEV STR SUB OPR ENQ Reason
  BO 0066 BATCH    *      91  91   1   9   0   0   9   0   0   0   0   0 IYNXJ
          BATCH    1      92  57   1   5   0   0   5   0   0   0   0   0 IYNXJ
          BATCH    2      89  34   0   4   0   0   4   0   0   0   0   0 IYNXJ


   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE
   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE
```

- Note the Time towards the upper right corner. You can use F10 and F11 to scroll backwards and forwards through time.
- Note the Range. That is the number of seconds in the interval.
- On this page, the information covers from 07.08.20 to 07.10.00.

```
.                        RMF V1R12   Job Delays                     Line 1 of 3  .
.  Command ===>                                                  Scroll ===> CSR  .
.                                                                                 .
.  Samples: 100      System: MV23  Date: 05/28/12  Time: 07.08.20  Range: 100   Sec .
.                                                                                 .
.  Job: IYNXK          Primary delay: Job is waiting to use the processor.        .
.                                                                                 .
.  Probable causes: 1) Higher priority work is using the system.                  .
.                   2) Improperly tuned dispatching priorities.                   .
.                                                                                 .
.                                                                                 .
.  ------------------------- Jobs Holding the Processor -------------------------- .
.  Job:            IYNXJ         Job:         RH23MSTR         Job:         OMEGTEMS .
.  Holding:           4%         Holding:            2%         Holding:           1% .
.  PROC Using:        9%         PROC Using:         2%         PROC Using:        1% .
.  DEV Using:         0%         DEV Using:          0%         DEV Using:         0% .
.  --------------------------- Job Performance Summary --------------------------- .
.          Service       WFL -Using%- DLY IDL UKN ---- % Delayed for ---- Primary  .
.  CX ASID Class    P Cr  %   PRC DEV  %   %   %  PRC DEV STR SUB OPR ENQ Reason   .
.  BO 0066 BATCH    *      91   91   1   9   0   0   9   0   0   0   0   0 IYNXJ    .
.          BATCH    1      92   57   1   5   0   0   5   0   0   0   0   0 IYNXJ    .
.          BATCH    2      89   34   0   4   0   0   4   0   0   0   0   0 IYNXJ    .
.                                                                                 .
.                                                                                 .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE    .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE  .
```

- You can also overtype Time to get to the time you want.
- When you do that, keep an eye on Range. It might double. Overtype Range to get it back to the smaller Range.

```
                        RMF V1R12   Job Delays                        Line 1 of 3
 Command ===>                                                       Scroll ===> CSR

 Samples: 100      System: MV23  Date: 05/28/12  Time: 07.08.20  Range: 100   Sec

 Job: IYNXK          Primary delay: Job is waiting to use the processor.

 Probable causes: 1) Higher priority work is using the system.
                  2) Improperly tuned dispatching priorities.


 ------------------------- Jobs Holding the Processor --------------------------
 Job:            IYNXJ        Job:         RH23MSTR        Job:         OMEGTEMS
 Holding:           4%        Holding:           2%        Holding:           1%
 PROC Using:        9%        PROC Using:        2%        PROC Using:        1%
 DEV Using:         0%        DEV Using:         0%        DEV Using:         0%
 --------------------------- Job Performance Summary ---------------------------
         Service       WFL -Using%- DLY IDL UKN ---- % Delayed for ---- Primary
 CX ASID Class    P Cr  %   PRC DEV  %   %   %  PRC DEV STR SUB OPR ENQ Reason
 BO 0066 BATCH    *     91   91   1  9   0   0    9   0   0   0   0   0 IYNXJ
         BATCH    1     92   57   1  5   0   0    5   0   0   0   0   0 IYNXJ
         BATCH    2     89   34   0  4   0   0    4   0   0   0   0   0 IYNXJ


  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE
  F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE
```

- 07.08.20 is the 100 second interval before the MXT began. (MXT began right around 07.10.08. The suddenly higher CPU began at 07.09.58.)
- IYNXK is using 91% Processor and is delayed 9% for processor. IYNXJ is using 9% Processor.

```
.                          RMF V1R12   Job Delays                     Line 1 of 1  .
.  Command ===>                                                 Scroll ===> CSR    .
.                                                                                  .
.  Samples: 100      System: MV23  Date: 05/28/12  Time: 07.10.00  Range: 100   Sec .
.                                                                                  .
.  Job: IYNXK         Primary delay: Job is waiting to use the processor.          .
.                                                                                  .
.  Probable causes: 1) Higher priority work is using the system.                   .
.                   2) Improperly tuned dispatching priorities.                    .
.                                                                                  .
.                                                                                  .
.  ------------------------- Jobs Holding the Processor -------------------------- .
.  Job:            IYNXJ       Job:         OMPROUTE         Job:         DI23IRLM  .
.  Holding:           4%       Holding:           1%         Holding:           1%  .
.  PROC Using:       91%       PROC Using:        1%         PROC Using:        1%  .
.  DEV Using:         2%       DEV Using:         0%         DEV Using:         0%  .
.  --------------------------- Job Performance Summary --------------------------- .
.          Service       WFL -Using%- DLY IDL UKN ---- % Delayed for ---- Primary  .
.  CX ASID Class     P Cr  %   PRC DEV  %   %   %  PRC DEV STR SUB OPR ENQ Reason  .
.  BO 0066 BATCH     1     93   91   1  7   0   2    5   0   0   0   0   2 IYNXJ   .
.                                                                                  .
.                                                                                  .
.                                                                                  .
.                                                                                  .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE     .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE   .
```

- The 07.10.00 interval is mostly all in the MXT period. IYNXK hasn't changed much.
- But IYNXJ is using 91% processor. That is a lot more than the prior interval. Let's have a look at CPU.

```
.                      RMF Monitor III Primary Menu                 z/OS V1R12 RMF .
.  Selection ===> 3                                                                .
.                                                                                  .
.  Enter selection number or command on selection line.                            .
.                                                                                  .
.                                                                                  .
.    S SYSPLEX          Sysplex reports and Data Index                        (SP) .
.    1 OVERVIEW         WFEX, SYSINFO, and Detail reports                     (OV) .
.    2 JOBS             All information about job delays                      (JS) .
.    3 RESOURCE         Processor, Device, Enqueue, and Storage               (RS) .
.    4 SUBS             Subsystem information for HSM, JES, and XCF          (SUB) .
.                                                                                  .
.    U USER             User-written reports (add your own ...)               (US) .
.                                                                                  .
.                                                                                  .
.                       O OPTIONS    T TUTORIAL    X EXIT                          .
.                                                                                  .
.         5694-A01 Copyright IBM Corp. 1986, 2010. All Rights Reserved             .
.                        Licensed Materials - Property of IBM                      .
.                                                                                  .
.                                                                                  .
.                                                                                  .
.                                                                                  .
.                                                                                  .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE     .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE   .
```

- Type ‘3’ for Selection and press ENTER.

```
.                      RMF Resource Report Selection Menu                          .
. Selection ===> 1A                                                                .
.                                                                                  .
. Enter selection number or command for desired report.                            .
.                                                                                  .
. Processor       1 PROC      Processor delays                   (PD)              .
.                1A PROCU     Processor usage                    (PU)              .
. Device          2 DEV       Device delays                      (DD)              .
.                 3 DEVR      Device resource                    (DR)              .
.                3A DSND      .. Data set level by DSN          (DSN)              .
.                3B DSNV      .. Data set level by volume       (DSV)              .
. Enqueue         4 ENQ       Enqueue delays                     (ED)              .
.                 5 ENQR      Enqueue resource                   (ER)              .
. Storage         6 STOR      Storage delays for each job        (SD)              .
.                 7 STORF     Storage usage by frames            (SF)              .
.                7A STORM     Storage usage by memory objects    (SM)              .
.                 8 STORR     Storage usage for each resource    (SR)              .
.                 9 STORS     Storage summary for each group     (SS)              .
.                10 STORC     Common storage summary             (SC)              .
.                11 STORCR    Common storage remaining          (SCR)              .
. I/O Subsystem  12 CHANNEL   Channel path activity              (CH)              .
.                13 IOQUEUE   I/O queuing activity               (IQ)              .
.                                                                                  .
.                                                                                  .
.  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE      .
.  F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE    .
```

- Type ‘1A’ for Selection and press ENTER.

```
.                        RMF V1R12   Processor Usage                    Line 1 of 21  .
.  Command ===>                                                       Scroll ===> CSR  .
.                                                                                      .
.  Samples: 100      System: MV23  Date: 05/28/12  Time: 07.08.20  Range: 100    Sec .
.                                                                                      .
.             Service     --- Time on CP % ---    ----- EAppl % -----                  .
.  Jobname  CX Class       Total    AAP     IIP        CP     AAP     IIP              .
.                                                                                      .
.  IYNXK    BO BATCH        94.0    0.0     0.0      94.0                              .
.  IYNXJ    BO IYNXJCLS      3.2    0.0     0.0       3.2                              .
.  WLM      S  SYSTEM        1.0    0.0     0.0       1.0                              .
.  XCFAS    S  SYSTEM        0.9    0.0     0.0       0.9                              .
.  RMFGAT   SO STC           0.8    0.0     0.0       0.8                              .
.  OMEGTEMS SO STCUSER       0.6    0.0     0.0       0.6                              .
.  GRS      S  SYSTEM        0.4    0.0     0.0       0.4                              .
.  NETVIEW  SO STCFAST       0.4    0.0     0.0       0.4                              .
.  OMEGCON  SO STC           0.3    0.0     0.0       0.3                              .
.  SMSVSAM  S  SYSTEM        0.1    0.0     0.0       0.1                              .
.  ZFS      S  SYSSTC        0.1    0.0     0.0       0.1                              .
.  JES2     S  STC           0.1    0.0     0.0       0.1                              .
.  RG23IRLM S  STC           0.1    0.0     0.0       0.1                              .
.  WJBMS41Z BO BATCH         0.1    0.0     0.0       0.1                              .
.  WJBCM41B BO BATCH         0.1    0.0     0.0       0.1                              .
.  WJBCM32B BO BATCH         0.1    0.0     0.0       0.1                              .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE        .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE      .
```

- With the Time set to 07.08.20, the interval before the MXT, we see that IYNXK was using most of a processor, and IYNXJ was using 3 percent of a processor.
- Press F11 to go to the next interval.

```
.                          RMF V1R12   Processor Usage                  Line 1 of 28  .
.  Command ===>                                                        Scroll ===> CSR   .
.                                                                                         .
.  Samples: 100      System: MV23  Date: 05/28/12  Time: 07.10.00  Range: 100    Sec .
.                                                                                         .
.             Service     --- Time on CP % ---     ----- EAppl % -----                    .
.  Jobname  CX Class       Total    AAP     IIP        CP     AAP     IIP                 .
.                                                                                         .
.  IYNXJ    BO IYNXJCLS     91.4    0.0     0.0      91.4                                 .
.  IYNXK    BO BATCH        87.6    0.0     0.0      87.6                                 .
.  DUMPSRV  S  SYSTEM        2.9    0.0     0.0       2.9                                 .
.  IXGLOGR  S  SYSTEM        1.1    0.0     0.0       1.1                                 .
.  XCFAS    S  SYSTEM        1.0    0.0     0.0       1.0                                 .
.  WLM      S  SYSTEM        1.0    0.0     0.0       1.0                                 .
.  RMFGAT   SO STC           0.8    0.0     0.0       0.8                                 .
.  OMEGTEMS SO STCUSER       0.6    0.0     0.0       0.6                                 .
.  GRS      S  SYSTEM        0.5    0.0     0.0       0.5                                 .
.  OMEGCON  SO STC           0.3    0.0     0.0       0.3                                 .
.  NETVIEW  SO STCFAST       0.3    0.0     0.0       0.3                                 .
.  CATALOG  S  SYSTEM        0.2    0.0     0.0       0.2                                 .
.  *MASTER* S  SYSTEM        0.1    0.0     0.0       0.1                                 .
.  RASP     S  SYSTEM        0.1    0.0     0.0       0.1                                 .
.  SMSVSAM  S  SYSTEM        0.1    0.0     0.0       0.1                                 .
.  CONSOLE  S  SYSTEM        0.1    0.0     0.0       0.1                                 .
.   F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=TOGGLE           .
.   F7=UP        F8=DOWN      F9=SWAP     F10=BREF     F11=FREF     F12=RETRIEVE         .
```

- During the MXT interval, IYNXJ is also using most of a processor.
- Could that cause transactions in IYNXK to use more CPU?

Systrace Perfdata

- Systrace Perfdata is an IPCS command that gives similar information to RMFIII regarding how much CPU is being used and what jobs are using it.
- Systrace Perfdata is new and newly documented at z/OS 1.12.
- We'll look at the SLIP dump triggered by the "Above_80_percent_of_MXT" message.
- We'll look at the dump of IYNXK taken after the problem was over, while it was doing its normal workload.

```
------------------------ IPCS Subcommand Entry ------------------------------
Enter a free-form IPCS subcommand or a CLIST or REXX exec invocation below:

===> systrace perfdata


----------------------- IPCS Subcommands and Abbreviations --------------------
ADDDUMP            | DROPDUMP, DROPD    | LISTDUMP, LDMP     | RENUM,    REN
ANALYZE            | DROPMAP,  DROPM    | LISTMAP,  LMAP     | RUNCHAIN, RUNC
ARCHECK            | DROPSYM,  DROPS    | LISTSYM,  LSYM     | SCAN
ASCBEXIT, ASCBX    | EPTRACE            | LISTUCB,  LISTU    | SELECT
ASMCHECK, ASMK     | EQUATE,   EQU, EQ  | LITERAL            | SETDEF,   SETD
CBFORMAT, CBF      | FIND,     F        | LPAMAP             | STACK
CBSTAT             | FINDMOD,  FMOD     | MERGE              | STATUS,   ST
CLOSE              | FINDUCB,  FINDU    | NAME               | SUMMARY,  SUMM
COPYDDIR           | GTFTRACE, GTF      | NAMETOKN           | SYSTRACE
COPYDUMP           | INTEGER            | NOTE,     N        | TCBEXIT,  TCBX
COPYTRC            | IPCS HELP, H       | OPEN               | VERBEXIT, VERBX
CTRACE             | LIST,     L        | PROFILE,  PROF     | WHERE,    W


 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=MORE     F7=UP
 F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=CURSOR
```

- This is on the SLIP dump.
- ENTER systrace perfdata

```
************************************************************* TOP OF DATA ********************************************

Note: Only SYSTRACE records available for ALL PROCESSORS are considered.

System: MV23 SP7.1.2 HBB7770


PERFDATA Analysis:

 CPU#    Went from          To             Seconds       SRB Time      TCB Time     Idle Time    CPU Overhead
 ---- ---------------- ---------------- ------------ ------------ ------------ ------------ ------------
  01  06:10:13.999836 06:10:14.912297     0.912460     0.008004     0.899153     0.000000     0.724603
  00  06:10:14.000223 06:10:14.912581     0.912358     0.005718     0.900525     0.000000     0.720400
                                        ------------ ------------ ------------ ------------ ------------
                                            1.824819     0.013722     1.799678     0.000000     1.445004

 SRB time       :      0.013722
 TCB time       :      1.799678
 Idle time      :      0.000000
 CPU Overhead   :      1.445004
                  ------------
         Total  :      1.824819

F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=RFIND   F6=MORE   F7=UP   F8=DOWN   F9=SWAP  F10=LEFT  F11=RIGHT
```

- Systrace Perfdata processes the system trace.
- We see that there are 2 processors doing work in the system trace.
- And each of those has trace covering about .9 seconds from 06:10:14.0 to 06:10:14.9.

```
****************************************************** TOP OF DATA ****************************************

Note: Only SYSTRACE records available for ALL PROCESSORS are considered.

System: MV23 SP7.1.2 HBB7770


PERFDATA Analysis:

 CPU#    Went from          To              Seconds      SRB Time     TCB Time     Idle Time    CPU Overhead
 ---- --------------- --------------- ------------ ------------ ------------ ------------ ------------
  01  06:10:13.999836 06:10:14.912297     0.912460     0.008004     0.899153     0.000000     0.724603
  00  06:10:14.000223 06:10:14.912581     0.912358     0.005718     0.900525     0.000000     0.720400
                                      ------------ ------------ ------------ ------------ ------------
                                          1.824819     0.013722     1.799678     0.000000     1.445004

 SRB time      :      0.013722
 TCB time      :      1.799678
 Idle time     :      0.000000
 CPU Overhead  :      1.445004
                  ------------
         Total :      1.824819

F1=HELP    F2=SPLIT    F3=END    F4=RETURN    F5=RFIND    F6=MORE    F7=UP    F8=DOWN    F9=SWAP   F10=LEFT   F11=RIGHT
```

- Idle Time of 0.00000 means that both processors were totally busy during the .9 seconds of systrace. There was never a moment when either had nothing to do.
- Use F8 to scroll down to see what jobs are using those 1.8 seconds of CPU time.

```
Found 54 address spaces in SYSTRACE.
Found 116 SRB and SSRB PSWs in SYSTRACE.

CPU breakdown by ASID:

ASID Jobname    SRB Time     TCB Time     Total Time
---- -------- ------------ ------------ ------------
0043 IYNXJ        0.001483     0.885384     0.886868
000B WLM          0.000609     0.009490     0.010100
0042 IYNXK        0.000498     0.863902     0.864401
0001 *MASTER*     0.000118     0.000309     0.000427
00A4 TCPIP        0.000824     0.000381     0.001206
009A RMFGAT       0.000020     0.013223     0.013244
0006 XCFAS        0.001730     0.003591     0.005322
0036 JES2MON      0.000400     0.000456     0.000856
00A2 IYCNCTGC     0.000024     0.000087     0.000111
009E C660CI23     0.000029     0.000079     0.000109
002E TN3270       0.000785     0.000244     0.001029
002F TN3270T2     0.000243     0.000253     0.000497
001C SMF          0.000577     0.000000     0.000577
00B9 RSED9        0.000014     0.000088     0.000102
00BE RSED7        0.000007     0.000032     0.000039
00B6 LOCKD        0.000006     0.000033     0.000040
F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=RFIND   F6=MORE   F7=UP   F8=DOWN
```

- Here we see that IYNXJ and IYNXK are together using up most of the 1.8 seconds of CPU time. They are each using most of a processor.
- Now let's take a look at the normal dump.

```
******************************************************** TOP OF DATA ****************************************

Note: Only SYSTRACE records available for ALL PROCESSORS are considered.

System: MV23 SP7.1.2 HBB7770


PERFDATA Analysis:

 CPU#    Went from           To               Seconds      SRB Time     TCB Time     Idle Time   CPU Overhead
 ---- --------------- ---------------  ------------ ------------ ------------ ------------ ------------
  00  06:15:23.897968 06:15:25.607760      1.709791     0.038181     0.900989     0.765688     0.718370
  01  06:15:23.906162 06:15:25.607608      1.701445     0.032215     0.895750     0.768634     0.751313
                                       ------------ ------------ ------------ ------------ ------------
                                           3.411237     0.070397     1.796739     1.534323     1.469683

 SRB time       :      0.070397
 TCB time       :      1.796739
 Idle time      :      1.534323
 CPU Overhead   :      1.469683
                  ------------
         Total :      3.411237

F1=HELP    F2=SPLIT    F3=END    F4=RETURN    F5=RFIND    F6=MORE    F7=UP    F8=DOWN    F9=SWAP   F10=LEFT   F11=RIGHT
```

- Here we see that each processor covers about 1.7 seconds of time.
- And we see there is significant Idle time, almost 1 processors worth of idle time.
- Scroll down to the next page.

```
Found 84 address spaces in SYSTRACE.
Found 207 SRB and SSRB PSWs in SYSTRACE.

CPU breakdown by ASID:

ASID Jobname    SRB Time     TCB Time     Total Time
---- -------- ------------ ------------ ------------
0042 IYNXK        0.000904     1.631235     1.632140
0043 IYNXJ        0.003651     0.044663     0.048315
0036 JES2MON      0.000765     0.000869     0.001634
009B DG23DBM1     0.000064     0.000083     0.000148
0001 *MASTER*     0.000252     0.000759     0.001011
0095 RMF          0.000137     0.001197     0.001334
001C SMF          0.001284     0.000000     0.001284
000B WLM          0.000965     0.021250     0.022216
00A4 TCPIP        0.001359     0.000625     0.001985
002C DI23MSTR     0.000237     0.000482     0.000719
002E TN3270       0.000619     0.000444     0.001063
002F TN3270T2     0.000384     0.000403     0.000787
0006 XCFAS        0.024349     0.001524     0.025873
0012 JESXCF       0.000406     0.000293     0.000700
0026 JES2         0.000087     0.000727     0.000815
0009 SMSVSAM      0.000448     0.001273     0.001721
F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=RFIND   F6=MORE   F7=UP   F8=DOWN
```

- IYNXK is using about 1 processors worth of CPU. And that is about it.
- So that squares with RMFIII. During the problem, IYNXJ and IYNXK are each using most of a processor. Before and after the problem, IYNXK is using about 1 processor and the other processor is pretty much idle.

And the answer is……

- It looks like the LPAR is about 50% busy when everything is fine. And it is 100% busy when the problem happens. Can that cause transactions to suddenly use 33% more CPU?
- Clues point us to IYNXJ. Let's take a look at the SMF110 data there to see what suddenly started using CPU.

Start Interval	Tran	#Tasks	Avg Response Time	Avg Suspend Time	Avg Dispatch Time	Avg User CPU Time	Total QR Disp Time	Avg QR Disp Time	Total QR CPU Time	Total KY8 Disp Time	Avg KY8 Disp Count	Total L8 CPU Time
07:08:11	CECI	1	245.4272	245.4141	.0131	.0046	.0131	.0131	.0046	.0000	0	.0000
07:09:58	SOAK	12	.0836	.0302	.0534	.0485	.0153	.0013	.0042	.6260	3	.5773
07:09:59	SOAK	19	.0771	.0241	.0530	.0484	.0171	.0009	.0061	.9897	3	.9129
07:10:00	SOAK	17	.0972	.0345	.0627	.0482	.0299	.0018	.0062	1.0355	3	.8134
07:10:01	SOAK	19	.0823	.0265	.0559	.0490	.0240	.0013	.0069	1.0377	4	.9240
07:10:02	SOAK	19	.0847	.0299	.0548	.0486	.0213	.0011	.0063	1.0202	4	.9172
07:10:03	SOAK	18	.0871	.0309	.0562	.0475	.0142	.0008	.0060	.9971	3	.8497
07:10:04	SOAK	19	.0796	.0257	.0539	.0486	.0234	.0012	.0062	1.0008	4	.9174

- This is a slightly tweaked DISPSUM form summarizing on 1-second intervals in IYNXJ.
- At exactly 07:09:58, SOAK transactions began.
- They are using a total of about .9 seconds of CPU per second, almost a whole processor. So that is why IYNXJ suddenly started using about 1 processors worth of CPU.

•The SOAK transaction does a loop of about 15 EXEC CICS GETMAIN followed by EXEC CICS FREEMAIN to get and free 20 Meg of EDSA, and it specifies INITIMG.

•INITIMG causes CICS, on every getmain, to write to every page of that 20 Meg.

•Part of the reason IYNXK transactions suddenly use more CPU is because the LPAR suddenly goes from 50% busy to 100% busy. At 50% busy as compared to 100% busy, the high-speed cache is more likely to always contain the pages of storage the instructions need. That is even more true given the fact that the SOAK transactions in IYNXJ are constantly writing to 20 Meg of storage. The constantly touching of the 20 Meg is making it so that the IYNXK transactions are constantly finding that the storage they need is not in the high-speed cache. That slows the IYNXK transactions down.

So what did you get?

- A neat new tool to put out console messages to expose MXT and near MXT
- A way to get a dump on MXT or near MXT
- A CICS Dispatcher refresher
- A way to approach response time spikes using SMF110 data
- A taste of how to make use of RMFIII
- A new IPCS tool: systrace perfdata
- An interesting reason why average CPU per transaction may vary from moment to moment

Questions and Answers