Session 17907

z/OS Debugging:
Diagnosing Loops & Hangs

z/OS Core Technologies – August 13th, 2015

Patty Little  plittle@us.ibm.com
John Shebey  jshebey@us.ibm.com
IBM Poughkeepsie

SHARE Orlando, August 2015
Trademarks

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

• MVS
• OS/390®
• z/Architecture®
• z/OS®

* Registered trademarks of IBM Corporation
Table of Contents

- Introduction .................................................. 4
- Diagnosing Loops ............................................. 10
- Loops and system trace ...................................... 12
- Finding status for looping work ............................. 19
- Diagnosing Hangs ............................................. 31
- IP ANALYZE RESOURCE .................................... 34
- Address space dispatchability ............................... 38
- Task dispatchability ......................................... 45
- Hangs and SRBs ............................................. 53
Introduction
What is a "Hang"?

Definition: No externally visible work being done by a process or function

Possible triggers

➤ Process is non-dispatchable
  ➤ May have no work to do
  ➤ May require a resource that is not available
  ➤ May be waiting for an event that is not occurring
➤ Process is dispatchable but not getting CPU
  ➤ May be a tuning problem
  ➤ May be that a higher priority address space is looping or consuming excessive CPU
➤ Process is looping
Documentation for diagnosis of loops & hangs

For address space loops and hangs:

- **Console Dump:**
  - **DUMP** **COMM=(name of your choice)**
  - **R x,JOBNAME=jjjjjjjj, SDATA=(RGN,CSA,LPA,SQA,ALLNUC,TRT,SUM,GRSQ),END**
    - Multiple jobnames may be specified

For system hangs:

- **Standalone Dump**

For either (and anything else you might debug!):

- **LOGREC**
- **SYSLOG/OPERLOG**
Or better yet, take advantage of RTD!

- Detects loop conditions such as HIGH CPU and TCB mode loops
- Detects hang conditions such as GRS and UNIX latch contention, ENQ contention, and local lock suspension

- START HZR, SUB=MSTR
- F HZR, ANALYZE, OPTIONS=(DEBUG=(LOOP))

- Produces a report of its findings (example on next slide)
- In the above example, if RTD detects a loop, the DEBUG option will cause it to automatically take a dump of the problem address space.
RTD report example

HZR0200I RUNTIME DIAGNOSTICS RESULT 581
SUMMARY: SUCCESS
REQ: 004 TARGET SYSTEM: SY1 HOME: SY1 2010/12/21 - 13:51:32
INTERVAL: 60 MINUTES
EVENTS:
FOUND: 02 - PRIORITIES: HIGH:02 MED:00 LOW:00
TYPES: HIGHCPU:01
TYPES: LOOP:01

EVENT 01: HIGH - HIGHCPU - SYSTEM: SY1 2010/12/21 - 13:51:33
ASID CPU RATE: 99% ASID:002E JOBNAME:IBMUSERX
STEPNAME:STEP1 PROCSSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MIGHT BE LOOPING.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.

EVENT 02: HIGH - LOOP - SYSTEM: SY1 2010/12/21 - 13:51:14
ASID:002E JOBNAME:IBMUSERX TCB:004FF1C0
STEPNAME:STEP1 PROCSSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE MIGHT BE IN A LOOP.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.

When both a HIGHCPU and a LOOP condition are detected by RTD, the Job is very likely looping.
Loop vs Hang in a dump

- IPCS SYSTRACE JOBNAME(j) TIME(LOCAL)

**LOOP**

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID</th>
<th>WU-Addr-</th>
<th>Ident</th>
<th>CD/D</th>
<th>PSW-----</th>
<th>Address-</th>
<th>Unique-1</th>
<th>Unique-2</th>
<th>Unique-3</th>
<th>Unique-4</th>
<th>Unique-5</th>
<th>Unique-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>TIMR</td>
<td>00000000_0767E656</td>
<td>00001005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07040000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>CLKC</td>
<td>00000000_0767F446</td>
<td>00001004 00000000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07040000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>0027</td>
<td>005F81A0</td>
<td>DSP</td>
<td></td>
<td>00000000_0767F446</td>
<td>00000000 07812870 08DCEA3C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07040000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>TIMR</td>
<td>00000000_0767E882</td>
<td>00001005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07042000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**HANG**

********** No Trace Table Entries meeting the selection criteria were found.**********
Diagnosing Loops
Steps for diagnosing a loop

- Goal is to locate a PSW and register set that can be used to “pump code” to explain the loop

- Identify loop pattern in system trace table
  - Note ASID and Work Unit
  - Note PSW addresses
  - Note environmental information
    - PSW ASC mode (P, S, H, or AR mode)
    - Cross memory environment (PASID, SASID)
    - Local lock status

- Use trace info to locate GPRs, ARs, and matching PSW for the looping unit of work
  - Use PSW address to identify looping code
  - Use regs to pump the code, determine reason for loop
Recognizing enabled loops in SYSTRACE

- Some loops are easy to pick out in the trace table by their repetitive pattern of events:

<table>
<thead>
<tr>
<th>Time</th>
<th>Function</th>
<th>Address</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PC</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>SSRV</td>
<td>132</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PR</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PC</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>SSRV</td>
<td>133</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PR</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PC</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>SSRV</td>
<td>132</td>
</tr>
<tr>
<td>0001</td>
<td>001A 008F8238</td>
<td>PR</td>
<td>0</td>
</tr>
</tbody>
</table>

- Other loops are a little trickier to recognize
Recognizing enabled loops in SYSTRACE

- **Characteristics of looping code**
  - Looping units of work are not doing productive work
    - Often not driving traceable events such as SVC and PC
  - Enabled looping units of work will be interrupted
    - Numerous I/O and EXTernal interrupts at similar PSW addresses
  - An interrupted TCB or preemptable SRB might get re-dispatched later on a different processor
  - An interrupted non-preemptable SRB will immediately be given back control on the same processor

- **How does this look in the system trace table?**
  - Many I/O, CLKC, and EXT trace entries; an occasional DSP
  - Similar PSW addresses
  - Same unit of work
  - Except for non-preemptable SRBs, loop may move across CPs
Example of a TCB mode loop

**IP SYSTRACE ASID(X’27’) TI(LO)**

| PR | ASID | WU-Addr | Ident | CD/D | PSW---- | Address-
|----|------|---------|-------|------|----------|----------
| 0001 | 0027 | 005F81A0 | EXT | TIMR | 00000000_0767E656 | 07040000_80000000
| 0001 | 0027 | 005F81A0 | EXT | CLKC | 00000000_0767F446 | 07040000_80000000
| 0002 | 0027 | 005F81A0 | DSP | 02002 | 00000000_0767E722 | 07040000_80000000
| 0002 | 0027 | 005F81A0 | EXT | TIMR | 00000000_0767E882 | 07042000_80000000

Work crosses CPs

Work Unit is TCB (addr is below line)

I/O, EXT, and DSP trace entries

Similar PSW addresses
Preemptable SRB mode loop

IP SYSTRACE ASID(X'25') TI(LO)

Like a TCB, a preemptable SRB can move across CPs.

The Work Unit address is above the line. It cannot be a TCB, so it must be a WEB. The WEB points to the SRB/SSRB.

Again we have EXT and I/O interrupt entries, but instead of DSP dispatch entries, we have SSRB dispatch entries.
Non-preemptable SRB mode loop

IP SYSTRACE ASID(X’26’) TI(LO)

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID WU-Addr-</th>
<th>Ident</th>
<th>CD/D PSW------ Address-</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002-0026 07F0BF00</td>
<td>EXT</td>
<td>CLKC 00000000_0767B89C 07041000 80000000</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>SSRV</td>
<td>110 810AEE00</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>EXT</td>
<td>TIMR 00000000_0767CA26 07040000 80000000</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>EXT</td>
<td>CLKC 00000000_0767B84E 07041000 80000000</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>SSRV</td>
<td>120 81360308</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>EXT</td>
<td>CLKC 00000000_0767BFD2 07042000 80000000</td>
<td></td>
</tr>
<tr>
<td>0002-0026 07F0BF00</td>
<td>I/O</td>
<td>0265E 00000000_0767C002</td>
<td></td>
</tr>
</tbody>
</table>

SRB stays on same CP.

Sometimes there is some “clutter” under the EXT trace entries. This is coming from code running under timer DIEs (disabled interrupt exits).

Work Unit address is that of a WEB.

Note there are no dispatch trace entries of any kind.
Simplifying the trace output

- With so many CPs (and therefore so much parallel activity) on some machines, it can be difficult to pick out a loop

- SYSTRACE offers several filtering options
  - TCB: SYSTRACE ASID(X’yy’) TCB(X’zzzzzzz’)
  - SRB: SYSTRACE WEB(X’zzzzzzzzz’)
  - Non-preemptable SRB
    SYSTRACE ASID(X’yy’) CPU(X’zzzzz’)
## Gathering info from SYSTRACE

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID</th>
<th>WU-Addr-</th>
<th>Ident</th>
<th>CD/D</th>
<th>PSW-----</th>
<th>Address-</th>
<th>PSACLHS-</th>
<th>PSALOCAL</th>
<th>PASD</th>
<th>SASD</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>TIMR</td>
<td>00000000_076CE8D6</td>
<td>00000000 00000000 0027 0027 17:22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07044000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>CLKC</td>
<td>00000000_076CE866</td>
<td>00000000 00000000 0027 0027 17:22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07047000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>0027</td>
<td>005F81A0</td>
<td>DSP</td>
<td></td>
<td>00000000_076CE866</td>
<td>00000000 00000000 0027 0027 17:22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07047000 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>0027</td>
<td>005F81A0</td>
<td>EXT</td>
<td>TIMR</td>
<td>00000000_076CE834</td>
<td>00000000 00000000 0027 0027 17:22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07044001 80000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Note ASID and WU (Work Unit) address** (TCB/WEB)
- **If non-preemptable SRB, note Processor number (CP)**
- **If TCB, note last bit of PSACLHS** (upper word)
  - If on, then work unit holds a local lock
  - **PSALOCAL** indicates ASCB whose local lock is held by this work unit
  - **PSALOCAL=0** indicates holding home address space’s local lock
Finding Status: TCB w/o lock

- SUMM FORMAT ASID(X’yy’)  [ASID from systrace]
- FIND ‘TCB: 00zzzzzz’  [TCB from systrace]

- General Purpose Registers
  - TCB, under heading “64-Bit GPRs from TCB/STCB”

- Access Registers
  - TCB’s STCB+X’30’

- PSW
  - Current RB’s XSBOPS16 at XSB+X’F0’
    (Current RB is last one formatted under TCB)

- PASID and SASID
  - Current RB’s XSBPASID and XSBPSASID respectively
  - SASID: XSB+X’D6’  PASID: XSB+X’CE’
Registers in TCB/STCB

TCB: 005F81A0
- - - - - - - - - - - - - - - - - - - - - - - - - 12 LINE(S) NOT DISPLAYED

64-Bit GPRs from TCB/STCB
Left halves of all registers contain zeros
0-3  07812870  08DCE428  00000032  07812870
4-7  00000003  08DCF670  07630CC8  08DCF6A1
8-11 08DCF61D  08DCE8A8  076C806C  00000000
12-15 FFFFFFFF  08DCE8A8  876C7650  00000000
- - - - - - - - - - - - - - - - - - - - - - - - - 97 LINE(S) NOT DISPLAYED

STCB: 7FF80420
- - - - - - - - - - - - - - - - - - - - - - - - - 5 LINE(S) NOT DISPLAYED

+0030  AR0...... 005FDD40  AR1...... 00000000  AR2...... 00000000
+003C  AR3...... 00000000  AR4...... 00000000  AR5...... 00000000
+0048  AR6...... 00000000  AR7...... 00000000  AR8...... 00000000
+0054  AR9...... 00000000  AR10..... 00000000  AR11..... 00000000
+0060  AR12..... 00000000  AR13..... 00000000  AR14..... 00000000
+006C  AR15..... 00000000  LSSD..... 7FF82CA0  LSDP..... 7F54A138
- - - - - - - - - - - - - - - - - - - - - - - - - 5 LINE(S) NOT DISPLAYED

General purpose registers are formatted under the TCB, access registers under the STCB
### PSW and XMEM info in XSB

<table>
<thead>
<tr>
<th>TCB: 005F81A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 RBP...... 00FF040 PIE...... 00000000 DEB...... 00000000</td>
</tr>
<tr>
<td>- - - - - - - - - - - - - - - - - - 217 LINE(S) NOT DISPLAYED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRB: 005FF040</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0020 XSB...... 7FFFDCE0 FLAGS2... 80 RTPSW1... 00000000</td>
</tr>
<tr>
<td>- - - - - - - - - - - - - - - - - - 29 LINE(S) NOT DISPLAYED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XSB: 7FFFDCE0</th>
</tr>
</thead>
<tbody>
<tr>
<td>- - - - - - - - - - - - - - - - - - 22 LINE(S) NOT DISPLAYED</td>
</tr>
<tr>
<td>+00CC KM....... 00C0 SASID.... 0027 PINS..... 00000006</td>
</tr>
<tr>
<td>+00D4 AX....... 0005 PASID.... 0027</td>
</tr>
<tr>
<td>- - - - - - - - - - - - - - - - - - 2 LINE(S) NOT DISPLAYED</td>
</tr>
<tr>
<td>+00F0 OPS16.... 07041000 80000000 00000000 0767E84E</td>
</tr>
</tbody>
</table>

- TCBRBP points to the “top” or “current” RB, which is the last RB formatted under this TCB in SUMM FORMAT.
- The corresponding XSB contains PSW and XMEM information.
Finding Status: TCB with lock

- If PSALOCAL non-zero:
  - CBF xxxxxx STR(ASCB)
  - FIND ASID
- SUMM FORMAT ASID(X’yy’)
  - If PSALOCAL=0: X’yy’ = Home ASID [ASID from systrace]
  - If PSALOCAL non-zero: X’yy’ is ASID found above
- FIND IHSA [field in ASXB]
- CBF X’zzzzzzz’ STR(IHSA) ASID(X’yy’)
  - PSW, GPRs, Ars
  - Note XSB address with IHSA
- CBF X’aaaaaaaaa’ STR(XSB) ASID(X’yy’)
  - PASID and SASID
### PSW, Regs in IHSA

<table>
<thead>
<tr>
<th>ASXB: 005FD820</th>
<th>IHSA: 005FE470</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0000 ASXB..... ASXB  FTCB..... 005FDD40  LTCB..... 005F81A0</td>
<td>+0000 CPUT..... 00000000  00770000  NTCB..... 005F81A0</td>
</tr>
<tr>
<td>+000C TCBS..... 0004  FLG1..... 00  SCHD..... 00</td>
<td>+000C OTCB..... 005F81A0  CPSW..... 070C0000  81529200</td>
</tr>
<tr>
<td>+0010 MPST..... 00000000  LWA..... 00000000  VFVT..... 00000000</td>
<td>- - - - - - - - - - - - - - - - - - - - - - - - - 3 LINE(S) NOT DI</td>
</tr>
<tr>
<td>+001C SAF..... 00000000  IHSA..... 005FE470  FLSA..... 00000027</td>
<td>General purpose register values</td>
</tr>
</tbody>
</table>

| 0-3  | D3D3D7E2 | 7FF59F50 | 0000002B | 00000000 |
| 4-7  | 7FF7D46C | 01BC20F8 | 00000080 | 00FBB5C8 |
| 8-11 | 8123D180 | 7FF59F50 | 7FF7D400 | 00FDBF00 |
| 12-15| 01529240 | 81529200 | 8123D5C8 | 00F9A380 |

+0080 XSB..... 7FFFD430  FLGS..... 00

**Access register values**

| 0-3  | 00000000 | 00000000 | FFFFFFFF | FFFFFFFF |
| 4-7  | FFFFFFFF | FFFFFFFF | FFFFFFFF | FFFFFFFF |
| 8-11 | FFFFFFFF | FFFFFFFF | FFFFFFFF | FFFFFFFF |
| 12-15| FFFFFFFF | FFFFFFFF | 00000000 | 00000000 |
XMEM info in XSB

XSB: 7FFFD430
+0000  XSB...... XSB  LINK..... 00000000  XLIDR.... 00000000
- - - - - - - - - - - - - - - - - - - - - - - - - - 21 LINE(S) NOT DISPLAYED
+00CC  KM....... 00C0  SASID.... 0027  PINS..... 00000006
+00D4  AX....... 0000  PASID.... 0027
Finding Status: Preemptable SRB

- CBF X’xxxxxxxxxx’ STR(WEB)  [WEB address from systrace WU-addr]
- FIND UPTR

```plaintext
WEB: 0257B200
+0000 WEB ...... WEB FLAG1.... 0000 FLAG2.... 08
+0007 TYPE ...... 14 LOCK ...... 00000000 WUQP ...... 0772A600
+0010 CMAJOR_B. 00FE CMINOR_B. 0000 HASCB .... 00F9A380
+0018 UPTR .... 0269F020 UNEXT .... 0772A600 UPREV .... 00000000
```

- CBF X’yyyyyyyyy’ STR(SRB)  [where yyyyyyyyy is UPTR value]
  - Note: STR(SRB) can be used for both SRB’s and SSRB’s
  - Locate PSW, GPRs, and ARs
  - FIND ‘XSB’  [to get address of XSB]
- CBF X’zzzzzzzzz’ STR(XSB)
  - Get PASID and SASID
### PSW, Regs in SSRB

<table>
<thead>
<tr>
<th>SSRB: 0269F020</th>
</tr>
</thead>
</table>

#### 64-Bit GPRs from SSRB/SSRX

**Left halves of all registers contain zeros**

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0–3 | 08186980 | 08BC38E4 | 00000000 | 00000000 |
| 4–7 | 0754D778 | 08BC574F | 08BC54C0 | 08BC674E |
| 8–11 | 08186970 | 08BC4750 | 076895BC | 0768A5BB |
| 12–15 | 00000000 | 08BC4750 | 87682884 | 00000000 |

**+0070 CPSW..... 07040000  8767E656**

**+0078 PSW16.... 07040000  80000000  00000000  0767E656**

**+0098 TOCP..... 00000013  254BF514**

**XSB...... 0269F0D0**

**+00A4 SSD...... 4F4F4F4F**

**SSRX..... 000001EF  83306000**

#### SSRX: 000001EF_83306000

**Access register values**

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0–3 | 00000000 | 00000000 | 00000000 | 00000000 |
| 4–7 | 00000000 | 00000000 | 00000000 | 00000000 |
| 8–11 | 00000000 | 00000000 | 00000000 | 00000000 |
| 12–15 | 00000000 | 00000000 | 00000000 | 00000000 |
### PSW, Regs, XMEM info in XSB

**XSB: 0269F0D0**

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 21 LINE(S) NOT DI

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+00C0</td>
<td>00000000 00000000</td>
<td>TRNE..... SINS.....</td>
</tr>
<tr>
<td>+00CC</td>
<td>8000</td>
<td>KM.......</td>
</tr>
<tr>
<td>+00D4</td>
<td>0005</td>
<td>AX.......</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SASID.... 0026</strong> <strong>PASID.... 0026</strong></td>
</tr>
</tbody>
</table>
Finding Status: Non-preemptable SRB

- CBF PSAx  [where x is Processor from systrace]
- FIND SCFS
- CBF X’yyyyyyyyy’ STR(SCFS)  [where yyyyyyyyy is SCFS addr]
  - GPRs in SCFSX1G0
  - Access Registers in SCFSX1A0
  - PSW in SCFSP161
  - Cross memory environment in SCFSX1SS (SASID) and SCFSX1PS (PASID)
**PSW, Regs, XMEM info in SCFS**

<table>
<thead>
<tr>
<th>SCFS: 020FC100</th>
</tr>
</thead>
<tbody>
<tr>
<td>+00D0 X1G0..... 0819B088 X1G1..... 08D66460 X1G2..... 00000024</td>
</tr>
<tr>
<td>+00DC X1G3..... 07627D60 X1G4..... 07627DF8 X1G5..... 0819B088</td>
</tr>
<tr>
<td>+00E8 X1G6..... 07630F84 X1G7..... 00000000 X1G8..... 07627DF8</td>
</tr>
<tr>
<td>+00F4 X1G9..... 08D666B0 X1GA..... 076C806C X1GB..... 00000000</td>
</tr>
<tr>
<td>+0100 X1GC..... 08D674C8 X1GD..... 08D666B0 X1GE..... 0771434E</td>
</tr>
<tr>
<td>+010C X1GF..... 076BE930</td>
</tr>
<tr>
<td>+0110 X1A0..... 00000000 X1A1..... 00000000 X1A2..... 00000000</td>
</tr>
<tr>
<td>+011C X1A3..... 00000000 X1A4..... 00000000 X1A5..... 00000000</td>
</tr>
<tr>
<td>+0128 X1A6..... 00000000 X1A7..... 00000000 X1A8..... 00000000</td>
</tr>
<tr>
<td>+0134 X1A9..... 00000000 X1AA..... 00000000 X1AB..... 00000000</td>
</tr>
<tr>
<td>+0140 X1AC..... 00000000 X1AD..... 00000000 X1AE..... 00000000</td>
</tr>
<tr>
<td>+014C X1AF..... 00000000 XRSA..... 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0250 P161..... 07046001 80000000 00000000 076C23E8</td>
</tr>
<tr>
<td>+03D0 X1SN..... 00000007 X1PK..... 8000 X1SS..... 0026</td>
</tr>
<tr>
<td>+03D8 X1PN..... 00000007 X1AX..... 0005 F1PS..... 0026</td>
</tr>
</tbody>
</table>

- - - - - - - - - - - - - - - - - - - - - - - - - - - 14 Line(s) not Display
- - - - - - - - - - - - - - - - - - - - - - - - - - - 15 Line(s) not Display
- - - - - - - - - - - - - - - - - - - - - - - - - - - 30 Line(s) not Display
- - - - - - - - - - - - - - - - - - - - - - - - - - - 78 Line(s) not Display
Pumping the code

- Do **IPCS WHERE** or **IPCS BROWSE** against the PSW address
  - If PSW address points to private storage, **make sure you specify the PASID as the ASID**
    - E.g. IP WHERE xxxxxxxx ASID(X’yy’)
- Before using your registers, check **PSW ASC mode bits** (bits 16 and 17)
  - 00 – Data reference is to primary address space
  - 10 – Data reference is using access registers
  - 01 – Data reference is to secondary address space
  - 11 – Data reference is to home address space
Diagnosing Hangs
Steps for diagnosing a hang

- Goal is to locate a unit of work that is the bottleneck, and to use its status (PSW, registers) and related information to explain why it is not progressing.

- Identify pivotal unit of work

- Gather dispatchability information about unit of work
  - If waiting, who did the WAIT?
  - If suspended, who did the SUSPEND?
  - If PAUSEd, who did the PAUSE?
  - If dispatchable, why isn’t it running?
System hangs

- System hang
  - Either the whole system, or else a major subset, is not functioning
  - Ideal documentation is a SADump but some diagnoses can be made using a console dump

- System hang diagnosis comes in 2 flavors: 😊 / 🙁
  - IP ANALYZE RESOURCE helps 😊
    - Report highlights contention and identifies the “bottlenecking” unit of work
  - IP ANALYZE RESOURCE doesn’t help 🙁
    - Need to work harder for answer
    - Consider what address spaces aren’t running
      - Verify their activity (or lack thereof) in system trace
        IP SYSTRACE JOBNAME(jjjjjjjjjj) TI(LO)
      - Explore their dispatchability
IP ANALYZE RESOURCE

- Use IP ANALYZE RESOURCE to identify contention
  - Identifies resource
  - Identifies owner and owner’s dispatchability status
  - Identifies contenders

- Report may call out multiple points of contention
  - Look for key system resources such as a local lock for a critical system address space
  - Look for long lists of contenders
  - Look for contention involving jobs you know to be hung

- **NOTE:** While ANALYZE RESOURCE is the “go-to” command for system hangs, it can be useful for address space hangs as well.
Details in ANALYZE RESOURCE report

- Examples of contention identified by ANALYZE RESOURCE
  - Suspend lock contention (LOCAL/CML/CMS)
    - Note: report calls this out even if no contenders
  - I/O device
  - ENQ resource (Major/Minor)
  - Page fault
  - Latches
    - Latch control blocks live in the latch set owner’s address space.
    - Contention will only show in ANALYZE RESOURCE if owner’s address space dumped.

- Examples of resource owner status identified by ANALYZE RESOURCE
  - Suspended or waiting
  - Interrupted but dispatchable
  - Executing on a CP
ANALYZE RESOURCE examples

RESOURCE #0004:
  NAME=LOCAL LOCK FOR ASID 00BA

RESOURCE #0004 IS HELD BY:
  JOBNAME=ABC  ASID=00BA  SSRB=1A31940C
  DATA=INTERRUPTED AND NOW DISPATCHABLE

RESOURCE #0002:
  NAME=LOCAL LOCK FOR ASID 0004

RESOURCE #0002 IS HELD BY:
  JOBNAME=*MASTER*  ASID=0001  SRB=00000000  CPU=26
  DATA=CURRENTLY RUNNING ON CPU 26

ASID 4=TRACE. Typical and not a concern in SVC dumps.
ANALYZE RESOURCE examples

RESOURCE #0002:
  NAME=MAJOR=CATLGRES MINOR=CAS SCOPE=SYSTEM

RESOURCE #0002 IS HELD BY:
  JOBNAME=CATALOG  ASID=0031  TCB=008AC680

RESOURCE #0002 IS REQUIRED BY:
  JOBNAME=PBLPROG  ASID=0117  TCB=008D1210

No dispatchability status; Perhaps CATALOG job was not in dump.
Checking address space dispatchability

- Check address space dispatchability if:
  - Problem is a hung address space
  - ANALYZE RESOURCE didn’t help identify the source of a system hang

- Steps for checking address space dispatchability
  - Check address space level non-dispatchability bits
  - Check task level non-dispatchability bits for key TCBs
    - Identifying key TCBs may require some inside knowledge of address space
    - For hangs during CANCEL or job shutdown, last TCB is often the bottleneck
  - Check RB level non-dispatchability indicators
  - Validate whether unit of work is on/off the WUQ dispatch queue

- What if address space is hung due to an SRB not running?
  - Locate SRB/SSRB on address space’s “in flight” queue
  - Validate whether unit of work is on/off the WUQ dispatch queue

Note: IP SUMM FORMAT ASID(X’yy’) to view ASCB/TCBs
ASCB non-dispatchability bits

Located in ASCBDSP1 at ASCB+X’72’, length 1 byte

Common settings:
- X'00' Address space is dispatchable
- X'80' Address space quiesced due to SVC dump in progress – not a problem!
- X'18' Contact Supervisor L2 (compID 5752SC1C5)
- X'40' Address space being terminated (MEMTERM)
- X'10' Address space logically swapped out
  OR
  TCB within address space has issued STATUS STOP of SRBs
Example: ASCB non-dispatchability

```
ASCB: 00F65D00
+0000  ASCB..... ASCB  FWDP..... 00F65B80  BWDP..... 00F65E80
+000C  LTCS..... 00000000  R010..... 00000000
+0018  IOSP..... 00000000  R01C..... 0000  WQID..... 0000
+0020  R020..... 00000000  ASID..... 007B
- - - - - - - - - - - - - - - - - - - - lines omitted - - - - - - - - -
+0068  TMCH..... 00000000  ASXB..... 008FD820  SWCT..... 004B
+0072  DSP1..... 10  FLG2..... 00
+0076  SRBS..... 0000  LLWQ..... 00000000  RCTP..... 008FDD40
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+0171  AVM2..... 00  AGEN..... 0000  ARC..... 00000000
+0178  RSMA..... 05885A30  DCTI..... 00000000

Address space non-dispatchability flags from ASCBDSP1:
STATUS stop SRB summary
```
Troubleshooting
ASCB non-dispatchability

X'40'
Address space being terminated (MEMTERM)

- Memterm is usually a quick process ... SO ...
  - Address space in memterm processing => memterm hung

- To debug:
  - Get a console dump of ASID1
    (memterm is driven from ASID1)
  - IP SUMM FORMAT ASID(1)
  - FIND IEAVTMTR (to locate task driving memterm)
    - Eyecatcher appears under first RB belonging to IEAVTMTR TCB
    - Verify that Reg1 in first RB matches our ASCB address
      - If not, repeat FIND IEAVTMTR looking for other memterm TCBs
    - Apply TCB non-dispatchability checks against IEAVTMTR TCB
Troubleshooting ASCB non-dispatchability

- X’10’ Address space swapped out OR “STATUS STOP-ed”

- IP VERBX SRMDATA
  - FIND ‘ASID xxxx’ to locate swap status of addr space
  - If logically swapped, for what condition?
    - WAITing – Apply TCB non-dispatchability checks to key TCBs in address space
    - Unilaterally swapped – Possible MPL issue
  - If not logically swapped, then address space must be STATUS STOP-ed
Example: VERBX SRMDATA

<table>
<thead>
<tr>
<th>JOB</th>
<th>INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASID</td>
<td>007B</td>
</tr>
<tr>
<td>OUCB</td>
<td>04FCB500</td>
</tr>
</tbody>
</table>

**LS WAIT QUEUE**

- +10 (LSW) **LOGICALLY SWAPPED**
- +11 (PVL) **PRIVILEGED PROGRAM**
- +29 (SRC) **SWAP OUT REASON: LONG WAIT**

**ASCBSME**

- RAX ADDRESS IS 05885BA8
- SERVICE CLASS = SYSSTC
- WORKLOAD = SYSTEM
- INTERNAL CLASS= $SRMGOOD
- PERIOD = 01
Troubleshooting ASCB non-dispatchability

- **X'10'** Address space “STATUS STOP-ed”

  - A TCB in the address space has requested STATUS STOP of SRBs
    - z/OS stops all SRB and TCB work in this address space except for requesting TCB
  - To diagnose, we need to identify requesting TCB
    - All TCBs but one will have TCBSRBND bit on (TCB + X'AF', bit X'20' )
    - Apply TCB non-dispatchability checks to TCB with TCBSRBND off
Checking task dispatchability

- For key TCBs, or for each TCB in address space:
  - Is the TCB on the WUQ (dispatching queue)?
    IP IEAVWEBI WUQ
  - Yes, then why isn’t it running?
    - WUQ backed up?
      Further check IP IEAVWEBI WUQ
    - Dispatching priority issue?
      - How does our TCB’s priority compare to others on WUQ
        Further check IP IEAVWEBI WUQ
      - What ASIDs are running in system trace? (SYSTRACE ALL)
      - What is their dispatching priority compared to ours?
        (SUMM FORMAT ASID(X’yy’) ; F DPH [within ASCB] )
  - Is something looping?
    Check for loops in system trace (SYSTRACE ALL)
  - No, then check TCB non-dispatchability indicators
**IP IEAVWEBI WUQ report**

### SUMMARY BY WUQ. SORTED BY TOTL:

<table>
<thead>
<tr>
<th>WUQ@</th>
<th>TOTL</th>
<th>PROC</th>
<th>CPUMASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>0309A800</td>
<td>156</td>
<td>CP</td>
<td>FE000000 00000000</td>
</tr>
<tr>
<td>0309A200</td>
<td>2</td>
<td>CP</td>
<td>80000000 00000000</td>
</tr>
<tr>
<td>0309A000</td>
<td>2</td>
<td>ZIIP</td>
<td>01840000 00000000</td>
</tr>
</tbody>
</table>

There are actually multiple WUQs. Section shows number of ready-to-run work units on each WUQ; in this example, 156 indicates a busy system with some backup.

### SUMMARY BY ASID. SORTED BY TOTL:

| ASID | JOBNAME | TOTL | ZAAP | ... | PSRB | FSRB | EXIT | CMLP | ---- | -------- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ---- |
|------|---------|------|------|-----|-----|-----|------|------|------|------|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0254 | WXYZ | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

WXYZ has 8 ready SRBs (across all WUQs)

### DETAILED INFORMATION FOR WUQ 0309A200, SORTED BY WEB DPH:

<table>
<thead>
<tr>
<th>WEB@</th>
<th>TYPE</th>
<th>WEB DPH</th>
<th>WUQ@</th>
<th>CLAS</th>
<th>ASCB@</th>
<th>PROMOTE</th>
<th>JOBNAME</th>
<th>ASID</th>
<th>DPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>045C3100</td>
<td>SRB</td>
<td>0FF8000</td>
<td>0309A200</td>
<td>CP</td>
<td>00FA5200</td>
<td>02157FA0</td>
<td>00000000</td>
<td>XCFAS</td>
<td>0006</td>
</tr>
<tr>
<td>1390BE80</td>
<td>TCB</td>
<td>00F240FF</td>
<td>0309A200</td>
<td>CP</td>
<td>00F12480</td>
<td>008A2CF0</td>
<td>04000000</td>
<td>W1234567</td>
<td>01E1</td>
</tr>
</tbody>
</table>

TCB’s address is omitted - - - - - - - - -
Before we start checking the TCB...

- As with a loop, we will be looking for status information (PSW, regs, xmem environment)

- Reminder:
  - **Status for an unlocked TCB** is saved as follows:
    - PSW in top RB’s XSBOPS16
    - GPRs in TCB; ARs in STCB
    - XMEM info (PASID, SASID) in XSB pointed to by TCB
  - **Status for a locally locked TCB** is saved in the IHSA of the address space whose lock is held
    - PSW, GPRs, and ARs in IHSA
    - XMEM info in XSB pointed to by IHSA
  - **Instruction execution** occurs in the **primary** address space

- If a TCB is locally locked
  - TCBLLH (+X’114’, X’01’ bit) on
  - TCBXLAS (+X’E8’) holds addr of ASCB whose lock is held
TCB non-dispatchability bits

Located in TCB at:
TCBFLGS (+1D) – last two bytes of 5-byte field
TCBNDSP (+AC) – 4 bytes

Common settings:

- FLGS = xxxxxx04 01  Top RB in a wait, check TCBNDSP
  NDSP = 00002000  Task non-dispatchable for SVCDump

- FLGS = xxxxxx00 01  Check TCBNDSP
  NDSP = 00002000  Task non-dispatchable for SVCDump

- FLGS = xxxxxx04 00  Top RB in a wait, TCBNDSP = 0 (SAdump)

- FLGS = xxxxxx00 00  No TCB non-dispatchability bits set

For other bit settings, see TCB mapping in MVS Data Areas

NOTE: TCB SUSPEND and PAUSE states are reflected elsewhere.
Checking TCB non-dispatchability

- Check non-dispatchability indicators formatted after TCB

TCB: 008FF6C8

+0000  RBP...... 008FF8D0  PIE...... 00000000  DEB...... 00000000
+000C  TIO...... 00D69FD0  CMP...... 00000000  TRN...... 40000000
+0018  MSS...... 7FFFE748  PKF...... 00  FLGS...... 00008004 01
+0022  LMP...... FF  DSP...... FF
- - - - - - - - - - - - - lines omitted - - - - - - - - - - - - - - - - - - -
+00AC  NDSP..... 00002000  MDIDS.... 00000000  JSCB..... 008FCE84
+00B8  SSAT..... 008FC390  IOBRC.... 00000000  EXCPD.... 00000000
- - - - - - - - - - - - - lines omitted - - - - - - - - - - - - - - - - - - -
+0154  SENV..... 008FC168

Task non-dispatchability flags from TCBFLGS4:

**Top RB is in a wait**

Task non-dispatchability flags from TCBFLGS5:

**Secondary non-dispatchability indicator**

Task non-dispatchability flags from TCBNDSP2:

**SVC Dump is executing for another task**

Why? Check top RB’s PSW

Just means additional bits on in TCBNDSP (per below)

Ignore; result of this dump
If TCB is waiting....

- Get PSW from XSBOPS16 of top RB’s XSB
- IP WHERE or IPCS Browse the PSW address, making sure you use the correct PASID
  - IEAVEWAT? Then WAIT was PC-entered
    - Find last LSE linkage stack entry (between TCB and STCB)
    - LSE TARG field should contain 0000030D
    - LSE PSWE will point to who issued the PC 30D WAIT
  - Otherwise, WAIT was SVC- or branch-entered
    - XSBOPS16 points to the issuer of the WAIT
TCB suspended?

Is TCB suspended?

- Get first byte of RBLINK from TCB’s top RB
  - If X’01’ then TCB is suspended
    - If TCB is unlocked, get suspend PSW from top RB’s XSBOPS16
    - If TCB is locked, get suspend PSW from IHSACPSW
    - IP WHERE or IPCS Browse the PSW address, making sure to use
      the correct PASID

NOTE: The SUSPEND PSW typically points right after a BALR
instruction. If this is not the case, the SUSPEND could be due to a
translation exception (e.g. page fault). Check for a non-zero
XSBRTRNE (XSB+X’C0’) that matches the instruction base register
or the instruction address.

X’01’ can also mean TCB is waiting, but we’ve already verified that this is not the case.
TCB not dispatchable: other checks

- Get PSW from XSBOPS16 of top RB’s XSB or from IHSACPSW
- IP WHERE or IPCS Browse the PSW address, making sure to use the correct PASID
  - IEAVEPS1?  => TCB is PAUSEd
    - Reg13 from TCB points to standard register save area
    - Reg13+C contains address where PAUSE was issued (use PASID when mapping return address to code)
  - IEAVESLK?  => TCB is suspended for a lock
    - Reg14 from TCB/IHSA indicates caller
    - Did you check IP ANALYZE RESOURCE?
Address space hang due to SRB/SSRB

Sometimes a “missing” SRB/SSRB causes an address space to hang

- Address space has a queue of in-flight SRBs
  - May be dispatchable (e.g. on WUQ) – **IP IEAVWEBI WUQ**
  - May be delayed/suspended for local lock – **ANALYZE RESOURCE**
  - May be suspended for page fault or other translation
  - May be WAITing
  - May be suspended explicitly by owner (SSRB)
  - May be PAUSEd

- Use **IP IEAVWEBI SRB ASID(xx)** to format an address space’s “in flight” SRBs and SSRBs
  - Use SRBEPA to recognize “missing” SRB
  - Need to do: **CBF ssrbaddr STR(SRB)** to get SSRB’s EPA
**Example: IEAVWEBI SRB ASID(xx)**

### Detailed Information for Webs on ASCB Web Queue:

<table>
<thead>
<tr>
<th>DATA FROM WEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB@</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>3C4C2300</td>
</tr>
<tr>
<td>13E6BB80</td>
</tr>
<tr>
<td>03824280</td>
</tr>
<tr>
<td>03A30E80</td>
</tr>
</tbody>
</table>

### ASCB Data

<table>
<thead>
<tr>
<th>JOBNAME</th>
<th>ASID</th>
<th>DPH</th>
<th>EPA</th>
<th>PASID</th>
<th>PTB</th>
<th>LSDP</th>
<th>CPSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFGDIST</td>
<td>0140</td>
<td>00F2</td>
<td></td>
<td></td>
<td></td>
<td>03C4600</td>
<td>47740001 3EA48174</td>
</tr>
<tr>
<td>DEFGDIST</td>
<td>0140</td>
<td>00F2</td>
<td></td>
<td></td>
<td></td>
<td>06C9B600</td>
<td>47740000 015398B6</td>
</tr>
<tr>
<td>DEFGDIST</td>
<td>0140</td>
<td>00F2</td>
<td></td>
<td></td>
<td></td>
<td>0456BC68</td>
<td>47740001 3EA598C6</td>
</tr>
<tr>
<td>DEFGDIST</td>
<td>0140</td>
<td>00F2</td>
<td></td>
<td></td>
<td></td>
<td>03D07C68</td>
<td>47743001 3E70BEB8</td>
</tr>
</tbody>
</table>

**SRB/SSRB address**

**WUQ addr can be residual**

**Addr space disp priority**

**SRB's (original) entry point addr**

**Current linkage stack entry on suspend/interrupt**

**Suspend/interrupt PSW**
Non-dispatchable SSRBs

- Where does SSRBCPSW point?
  - IEAVSRBS?
    - PC Suspend
    - Get SSRBLSDP: IP CBF ssrblsdp-120 STR(LSE)
    - LSETARG field should contain 00000317; LSEPSWE points to caller
  - IEAVEWAT?
    - PC WAIT
    - Get SSRBLSDP: IP CBF ssrblsdp-120 STR(LSE)
    - LSETARG field should contain 0000030D; LSEPSWE points to caller
  - IEAVEPSS?
    - Paused.
    - SSRB Reg13 points to standard save area; +C is return address
Non-dispatchable SSRBs

Where does SSRBCPSW point? (cont)

- After a BALR?
  - Could be branch-entered SUSPEND
  - Check code where SSRBCPSW points for SUSPEND/CALLDISP macros

- Just a “regular instruction”? 
  - Could be suspended for a page fault or other translation exception
  - Get SSRXTRNE (SSRX formatted along with SSRB)
  - If non-zero, does it match base register or instruction address of instruction pointed to by SSRBCPSW?
Questions?