

Improve JES2 Performance Using Trace Records

SHARE Winter 2001, Session: 2658

Permission is granted to SHARE Inc. to publish this presentation in the SHARE proceedings. IBM retains its right to distribute copies of this presentation to whomever it chooses.

Author: Tom Wasik

Presenter: Chip Wood

JES2 Design/Development/Service

wasik@us.ibm.com

chipwood@us.ibm.com

- ▶ This presentation describes various JES2 trace records and how to use them to improve overall JES2 performance. It is assumed that you are familiar with JES2 and have a knowledge of the JES2 \$TRACE facility.
- ▶ This presentation was created using examples from the OS/390 release 10 level of JES2. Some differences may exist in older levels of JES2.

Overview

This presentation will cover the following:

- ▶ Quick overview of the JES2 \$TRACE facility
- ▶ Relevant trace record information
 - ID 17 - Checkpoint
 - ID 20 - SYSOUT work selection
 - ID 30 - Posting for new work
 - ID 31 - JOB work selection
 - Others
- ▶ Putting it all together

\$TRACE overview

- Formatted trace data written to SYSOUT data set
- TRACEDEF sets up environment
 - ▶ PAGES= and TABLES= set up ECSA buffers
 - PAGES= is init deck only
 - ▶ ACTIVE= turns tracing on and off (YES/NO)
 - ▶ LOG= controls SYSOUT data set
 - START=YES/NO controls writing to DS
 - CLASS= sets SYSOUT class
 - SIZE= Controls records before SPIN

\$TRACE overview

(continued)

- TRACE(n) controls individual IDs
 - ▶ START=YES/NO turns an ID on or off
- PCEs/Devices can have selective tracing
 - ▶ TR or TRACE=YES/NO controls device
- Trace data sets can be spun at any time
 - ▶ \$TTRACEDEF,SPIN

Turning on a trace and create SYSOUT

- \$T TRACEDEF,ACTIVE=YES,
LOG=(START=YES)

- \$STRACE(17)

run test

- \$PTRACE(17)

- \$T TRACEDEF,SPIN

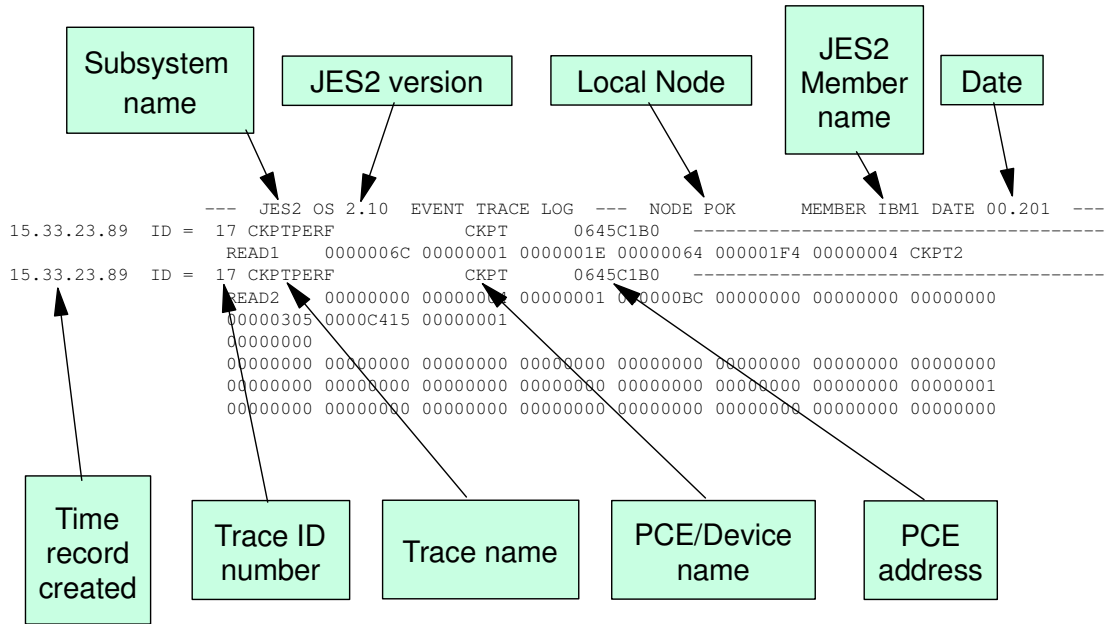
- \$T TRACEDEF,ACTIVE=NO

\$TRACE output

- SPIN SYSOUT data set created
 - ▶ JOBNAME is \$TRCLOG
 - ▶ This is an STC
 - ▶ SYSOUT class is as specified on TRACEDEF

```
$HASP686 OUTPUT($TRCLOG)  OUTGRP=1.1.1,BURST=NO,FCB=***,  
$HASP686                   FLASH=***,FORMS=STD,HOLD=(NONE),  
$HASP686                   OUTDISP=WRITE,PRIORITY=144,  
$HASP686                   PRMODE=LINE,QUEUE=A,  
$HASP686                   RECORDS=(91 OF 91),ROUTECD=LOCAL,  
$HASP686                   SECLABEL=,TSOAVAIL=NO,UCS=***,  
$HASP686                   USERID=IBMUSER,WRITER=
```

\$TRACE Output (continued)



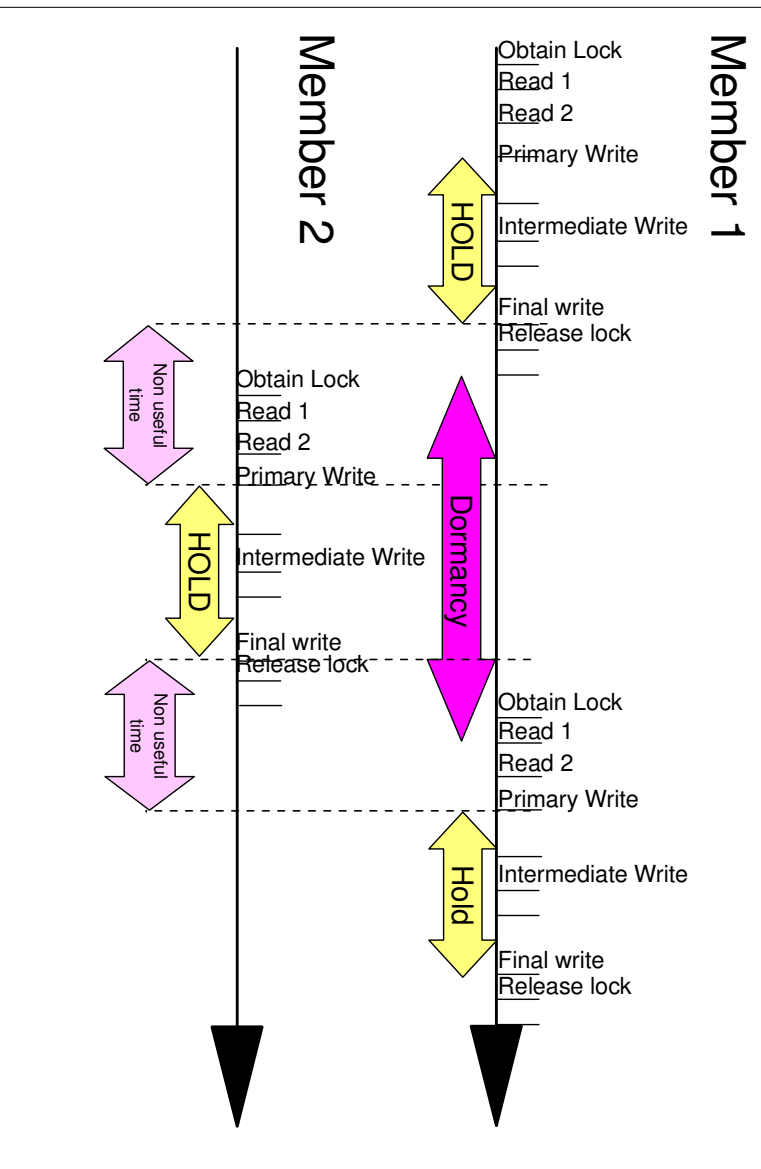
How can \$TRACE help me?

- Watch for unusual trace results
- Look for trends over time
- See how load affects trace data
- This implies looking at your system when there is no problem and setting up a baseline
- See how changes to the system affect trace output
- Look for problems before the effect is noticed

Trace 17 - Checkpoint

- This traces all CKPT I/O and reports various data
- Reduction program JES2T17A in SHASSAMP
- Most useful in an MAS
- Tuning knobs that this can help with
 - ▶ MASDEF HOLD= and DORMANCY=

The Checkpoint Cycle



Trace 17 - Checkpoint (*continued*)

- Goal in tuning
 - ▶ Reduce MAS performance penalty
 - ▶ Functions in MAS are delayed waiting for checkpoint
 - ▶ Impact of delay can be reduced by parallel processing
 - Parallel processing example - use multiple internal readers to submit jobs instead of a single internal reader
 - ▶ CKPT tuning can also reduce delays

Trace 17 - Checkpoint (*continued*)

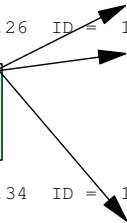
- Goals in tuning (continued)
 - ▶ Balancing act between
 - Reduce time to reacquire checkpoint
 - Hold long enough to not create additional delays
 - Keep non-useful time in CKPT cycle short
 - ▶ Trace 17 and its reduction program helps adjust the balance

Sample TRACE 17

```

16.15.00.25 ID = 17 CKPTPERF          CKPT      0645C1B0 -----
                READ1  0000006E 00000003 0000001E 00000064 000001F4 00000004 CKPT2
16.15.00.26 ID = 17 CKPTPERF          CKPT      0645C1B0 -----
                READ2  00000000 00000004 00000087 000000BC 00000006 000027AD 000023B7
                00002F9C 00002886 0000000C
                0053B510
                00000000 0000000A 00000000 00000000 00000000 00000006 00000000 0000001A
                00000000 00000014 00000000 00000007 00000000 0000000E 00000000 00000002
                00000000 00000032 00000000 00000000 00000000 00000000 00000000 00000000
16.15.00.34 ID = 17 CKPTPERF          CKPT      0645C1B0 -----
                PRIMARY 00000342 00000000 00000000 000000BC 00000001 0000030E 0000030E
                00000000          00000000 0000000C 00004265 CKPT1
                00000000 00000000 00000000
                00000001 00000000 00000000 00000000 00000001 00000000 00000002 00000000
                00000002 00000000 00000001 00000000 00000001 00000000 00000001 00000000
                00000003 00000000 00000000 00000000 00000000 00000000 00000000 00000000
16.15.00.38 ID = 17 CKPTPERF          CKPT      0645C1B0 -----
                INTERMED 000001B7 00000002 0000002C 000000BC 00000001 000000C8 000000C8
                000011AB          00000001 0000000C 00004266 CKPT1
                0000009D 00013605 00000ED6
                00000000 00000002 00000000 00000000 00000000 00000002 00000000 00000009
                00000000 00000006 00000000 00000002 00000000 00000005 00000000 00000001
                00000000 00000011 00000000 00000000 00000000 00000000 00000000 00000000
16.15.00.62 ID = 17 CKPTPERF          CKPT      0645C1B0 -----
                FINAL   000000AD 00000002 00000010 000000BC 00000000 00000000 00000000
                00002406 00000DCB          00000001 00000007 00004268 CKPT1
                0000012B 00044BFB 0000274D
                00000000 00000004 00000000 00000000 00000000 00000004 00000000 00000012
                00000000 0000000E 00000000 00000004 00000000 0000000B 00000000 00000002
                00000000 00000021 00000000 00000000 00000000 00000000 00000000 00000000
    
```

Record Type



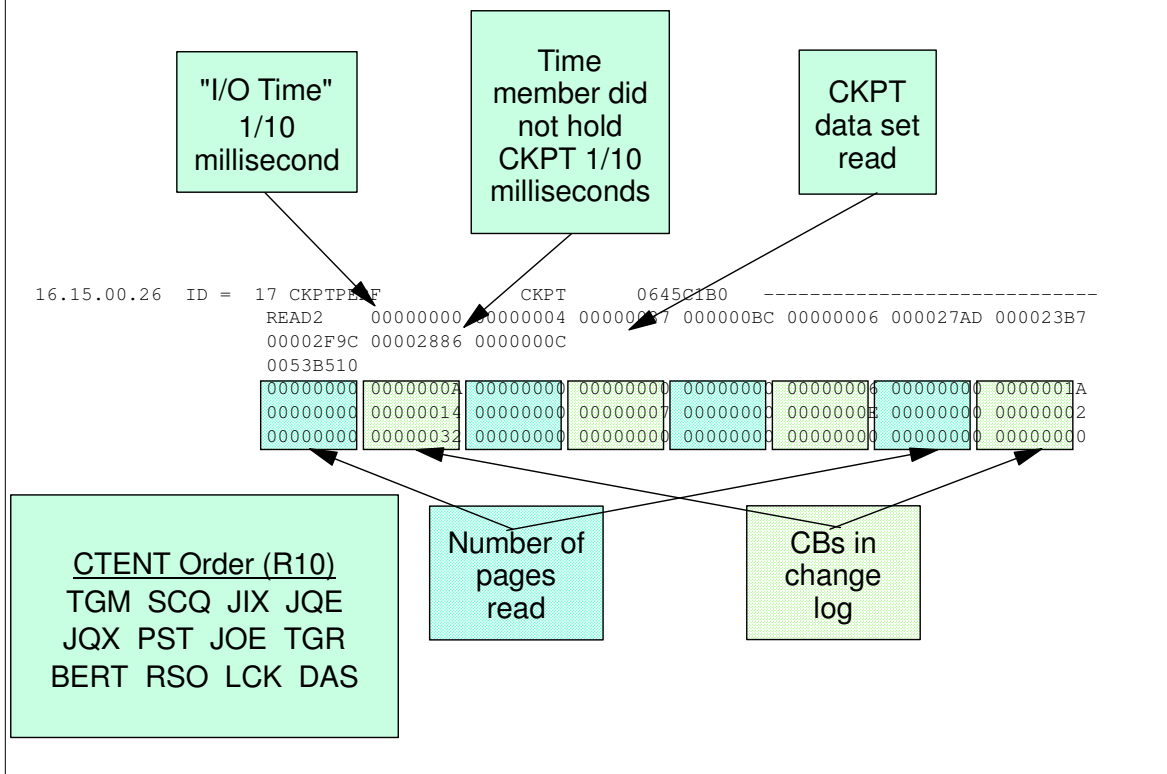
Sample TRACE 17 - READ1

```
16.15.00.25 ID = 17 CKTPERF          CKPT      0645C1B0 -----  
READ1      0000006E 00000003 0000001E 00000064 000001F4 00000004 CKPT2
```

"I/O Time"
1/10
millisecond

CKPT
data set
read

Sample TRACE 17 - READ2



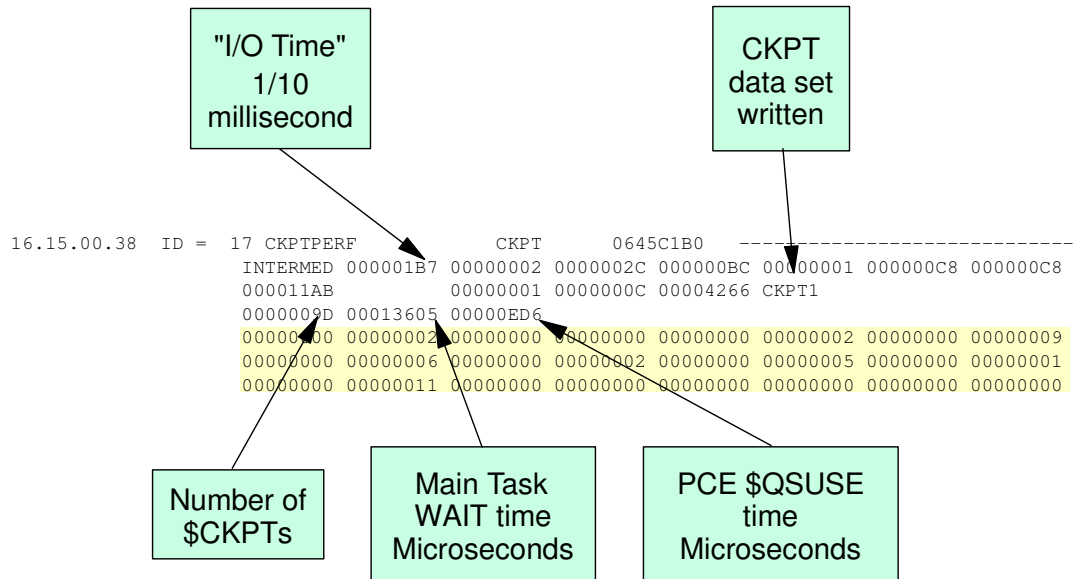
Sample TRACE 17 - PRIMARY

"I/O Time"
1/10
millisecond

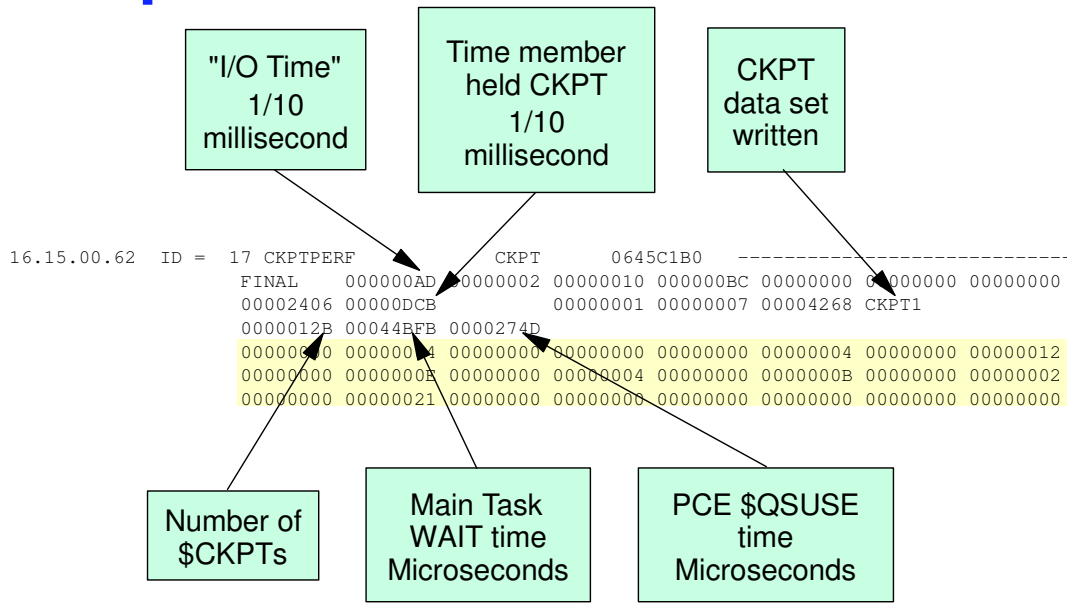
CKPT
data set
written

```
16.15.00.34 ID = 17 CKPTPERF          CKPT      0645C1B0 -----  
PRIMARY 00000342 00000000 00000000 000000BC 00000001 0000030E 0000030E  
00000000          00000000 0000000C 00004265 CKPT1  
00000000 00000000 00000000  
00000001 00000000 00000000 00000000 00000001 00000000 00000002 00000000  
00000002 00000000 00000001 00000000 00000001 00000000 00000001 00000000  
00000003 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```


Sample TRACE 17 - INTERMED



Sample TRACE 17 - Final



Trace 17 Reduction Program

- Summarizes trace 17 data
 - ▶ JES2T17A in SHASSAMP
 - ▶ Normal way to look at trace data
 - ▶ Useful for trend analysis
 - ▶ See prolog for how to run

Trace 17 Reduction Program

Non-zero counts means data was read and processed

```

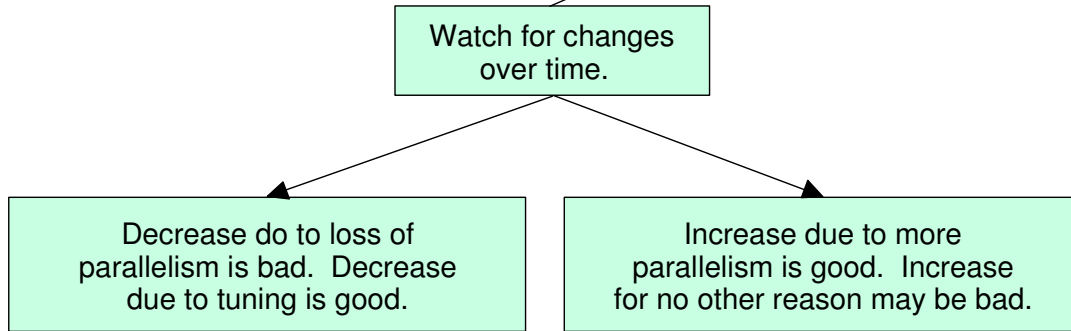
TRACE DATASET STATISTICS
=====
TOTAL NUMBER OF TRACE RECORDS READ:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ - READ1:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ - READ2:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ - PRIMARY:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ - INTERMEDIATE:
TOTAL NUMBER OF CHECKPOINT (TYPE 17) TRACE RECORDS READ - FINAL:
TOTAL NUMBER OF TRACE FILE ERRORS:
JES2 NODE NAME:
JES2 MEMBER NAME:
START DATE OF TRACE:
TIME STAMP - FIRST TRACE RECORD (START OF TRACE):
TIME STAMP - LAST TRACE RECORD (END OF TRACE):
CHECKPOINT CYCLE STATISTICS
=====
TOTAL          AVERAGE          MINIMUM          MAXIMUM
=====          =====          =====          =====
CYCLE TIME - (READ2 TO READ2):
107640 MS      1416 MS          1330 MS          1710 MS
    
```

Cycle time good first place to look
Under 2000 is very good

High MAX could indicate a problem

Trace 17 Reduction Program

READ2 STATISTICS	TOTAL	AVERAGE	MINIMUM	MAXIMUM
=====	=====	=====	=====	=====
NUMBER OF PCES \$WAITING FOR ACCESS TO CHECKPOINT:	405	5	2	9
MAX TIME PCE \$WAITING FOR ACCESS TO CHECKPOINT:	78226 MS	1003 MS	12 MS	1221 MS
AVG TIME PCE \$WAITING FOR ACCESS TO CHECKPOINT:	70986 MS	910 MS	10 MS	1115 MS



Trace 17 Reduction Program

=====

COMPARISON OF AVERAGE ELAPSED TIMES FOR CKPT1 AND CKPT2:

=====

	CKPT1	CKPT2
AVERAGE ELAPSED I/O TIME FOR READ1:	20 MS	16 MS
AVERAGE ELAPSED I/O TIME FOR READ2:	0 MS	0 MS
AVERAGE ELAPSED I/O TIME FOR PRIMARY:	26 MS	49 MS
AVERAGE ELAPSED I/O TIME FOR INTERMEDIATE:	37 MS	25 MS
AVERAGE ELAPSED I/O TIME FOR FINAL:	28 MS	30 MS

Watch for I/O time change not related to workload change

=====

COMPARISON OF TIME THIS SYSTEM HELD THE CHECKPOINT VERSUS INITIAL SETTING FOR MASDEF HOLD:

=====

AVERAGE TIME THIS SYSTEM HELD THE CKPT:	330 MS
INITIAL SETTING OF MASDEF HOLD:	30 HUNDRETHS SECONDS

=====

COMPARISON OF TIME THIS SYSTEM DID NOT HOLD THE CHECKPOINT VERSUS INITIAL MINIMUM SETTING FOR MASDEF DORMANCY:

=====

AVG TIME THIS SYS DID NOT HOLD CKPT:	1114 MS
INITIAL MINIMUM SETTING OF MASDEF DORMANCY:	100 HUNDRETHS SECONDS

These are the actual hold and dormancy (usually larger than parameter).

Trace 20 - SYSOUT work selection

- Traces most calls to select SYSOUT for processing
 - ▶ Printer/Punch, SAPI, NJE, RJE, Offload
 - ▶ NOT PSO, External Writer
- Displays selection criteria
- Reports overhead of selection
- Used to determine efficiency of setup
- Tuning knobs that this can help with
 - ▶ WS= on devices
 - ▶ SYSOUT processing philosophy

Sample TRACE 20

How many output elements we look at in some way

Name of Device

Elements we spent a lot of overhead examining


```

17.53.22.02 ID = 20 $#GET JOB00220 PRT2 06464AB0 $#GET CALL FOR PRT2
WS = (W,Q,R,PRM,LIM/F,UCS,FCB)
OUTGRPS DEFINED = 200 OUTGRPS IN USE =
OUTGRPS SCANNED = 1 OUTGRPS THRU WS = 1
OUTGRP MASK = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFF
CLASS = A ROUTE = 00010000 FLAGS = 20A08000
FAST EXIT INDICATOR SET - FAST EXIT SUCCESSFUL
ELEMENT SELECTED = 1
CPU TIME USED (SEC) = 0.000013
$#GET CALLED BY = HASPPRP 000A6590 + 0009F2
    
```

New in R10, if fast path is set, then other data in this record is not useful for tuning. No mater how bad the selection was, the overhead is always 1 element. More Fast paths are much better.

Closer element selected to the number scanned the better

Sample TRACE 20

2 characters in Outgrp mask correspond to one element in WS=. Priority counts as 2 places.

```

17.55.48.42 ID = 20 $#GET      JOB00286  PRT2      06464AB0  $#GET CALL FOR PRT2
WS = (W,Q,R,PRM,LIM/F,UCS,FCB)
OUTGRPS DEFINED =          200  OUTGRPS IN USE      =          77
OUTGRPS SCANNED =           2  OUTGRPS THRU WS    =           1
OUTGRP MASK = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFF
CLASS = A  ROUTE = 00010000  FLAGS = 20A00000
ELEMENT SELECTED =          2
CPU TIME USED (SEC) =          0.000020
$#GET  CALLED BY = HASPPRPU 000A6590 + 0009F2
    
```

A value that is not FF indicates what WS value may have caused extra elements to be scanned. In this case, the printer was set to QUEUE=1A2B3C4D5E6F7G8 and the output selected was class A (second in the list).

Sample TRACE 20

These entries actually selected output. However, do not forget to look at entries that do NOT select any output (Element selected = 0). The cost of these are often greater than the cost to select output.

```

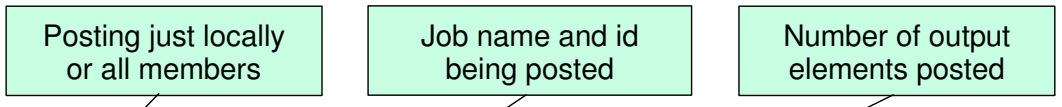
17.55.48.42 ID = 20 $#GET      JOB00286   PRT2      06464AB0  $#GET CALL FOR PRT2
            WS = (W,Q,R,PRM,LIM/F,UCS,FCB)
            OUTGRPS DEFINED      =      200  OUTGRPS IN USE      =      77
            OUTGRPS SCANNED      =      2    OUTGRPS THRU WS     =      1
            OUTGRP MASK = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFF
            CLASS = A   ROUTE = 00010000  FLAGS = 20A00000
            ELEMENT SELECTED      =      2
            CPU TIME USED (SEC)   =      0.000020
            $#GET   CALLED BY    = HASPPRPU 000A6590 + 0009F2
    
```

CPU time gives a good estimate of actual cost to select work

TRACE 30 - \$#POST

- Work selection is a 2 part process
 - ▶ Devices selecting work to process
 - ▶ New and changed output finding devices to wake up
- Overhead on the posting side can be as high as the selection side
- When looking at device setup, look at both trace records to see the effect
- Tuning knobs that this can help with
 - ▶ Same as Trace 20
 - ▶ Command overhead

Sample TRACE 30



```

17.55.08.30 ID = 30 $#POST JOB00365 HOPE 0645A6A0 $#POST TYPE=JQE
MASPOST=YES
JOB = JOBT47 (JOB00365)
JOES SCANNED = 1
DEVICES SCANNED = 5 DEVICES POSTED = 0
PSO WRITERS SCANNED = 0 PSO WRITERS POSTED = 0
SAPI WRITERS SCANNED = 0 SAPI WRITERS POSTED = 0
WORK SELECTION CALLS = 0
CPU TIME USED (SEC) = 0.000014
$#POST CALLED BY = HASPHOPE 0005F0E0 + 000B06
    
```

Work selection calls are normal if devices are posted

CPU time gives a good estimate of actual cost to select work

Sample TRACE 30

For individual output element posts, the OUTGRP name is included

```

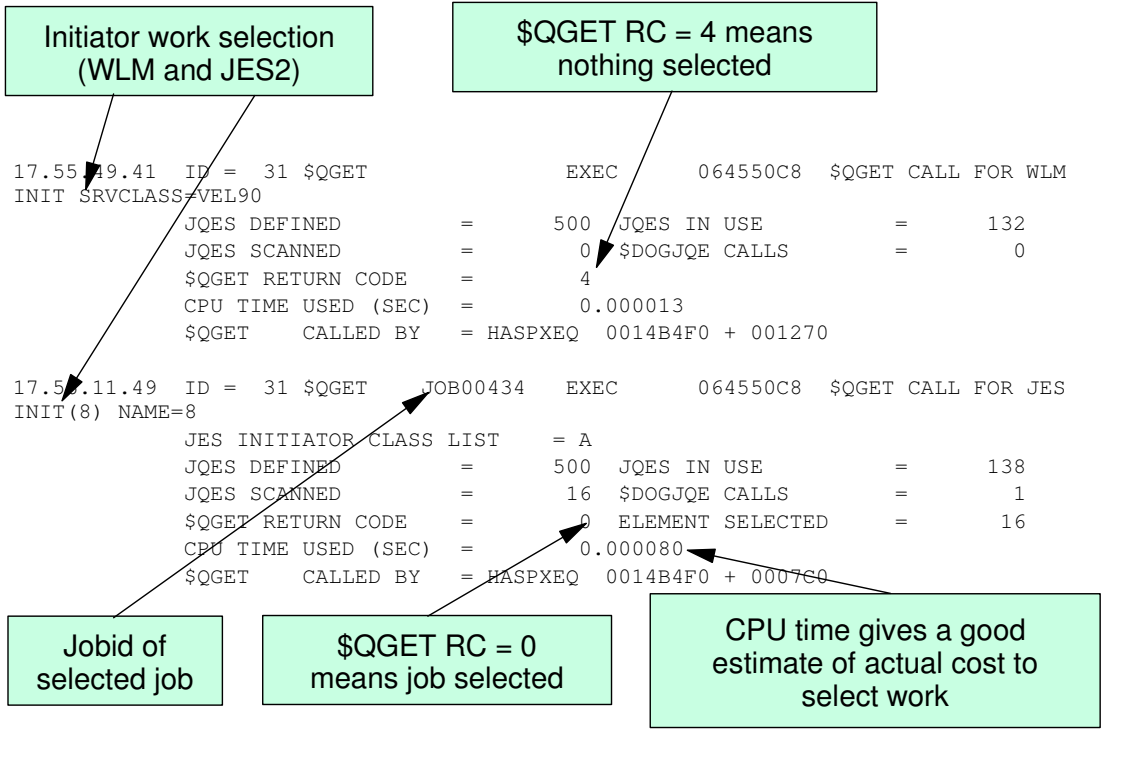
17.56.11.49 ID = 30 $#POST          CKPT      0645C1B0  ▼#POST TYPE=JOE
              JOB = JOBT77 (JOB00404)  ▲OUTGRP = 1.00001.00001
              DEVICES SCANNED = 5      DEVICES POSTED = 1
              PSO WRITERS SCANNED = 0    PSO WRITERS POSTED = 0
              SAPI WRITERS SCANNED = 0    SAPI WRITERS POSTED = 0
              WORK SELECTION CALLS = 1
              CPU TIME USED (SEC) = 0.000031
              $#POST CALLED BY = HASPCKPT 00028160 + 00139A
    
```

Entries from checkpoint are for cross member posts. Entries from HOPE and SPIN are new output. Entries from COMM are commands

Trace 31 - JOB selection

- Each phase of JES2 processing (other than input) must select work to process
- Offload devices select work to process
- Tuning knobs that this can help with
 - ▶ JES2 initiator class list
 - ▶ EXIT 14/49 performance impact

Sample TRACE 31



Sample TRACE 31

Not just for initiator selection

Overhead of extra idle processors is small (below what can be measured)

```

17.55.49 11 ID = 31 $QGET          CNVT      064557E0 $QGET CALL FOR
QUEUE=CNVT
          JQES DEFINED           =      500 JQES IN USE           =      132
          JQES SCANNED           =        1 $DOGJQE CALLS       =        0
          $QGET RETURN CODE      =         4
          CPU TIME USED (SEC)    =    0.000001
          $QGET CALLED BY       = HASPCNVT 0003E7F0 + 000102

17.56.10 25 ID = 31 $QGET      JOB00384  PURGE      06456080 $QGET CALL FOR
QUEUE=PURGE
          JQES DEFINED           =      500 JQES IN USE           =      145
          JQES SCANNED           =         4 $DOGJQE CALLS       =         1
          $QGET RETURN CODE      =         0 ELEMENT SELECTED   =         4
          CPU TIME USED (SEC)    =    0.000067
          $QGET CALLED BY       = HASPTRAK 0013FDA8 + 001AE6
    
```


Sample TRACE 31

```

19.24.19.79 ID = 31 $QGET JOB00008 EXEC 064550C8 $QGET CALL FOR JES
INIT(1) NAME=1
JES INITIATOR CLASS LIST = A
JQES DEFINED = 500 JQES IN USE = 18
JQES SCANNED = 1 $DOGJQE CALLS = 1
$QGET RETURN CODE = 0 ELEMENT SELECTED = 1
CPU TIME USED (SEC) = 0.000096
CPU TIME USED (X14) = 0.000004
EXIT 14 RETURN CODE = 0
CPU TIME USED (X49) = 0.000003
EXIT 49 SKIPPED JQES = 0
$QGET CALLED BY = HASPXEQ 0014B4F0 + 0007C0
    
```

Overhead of exits can be measured

Anything else?

- TRACE 27 - PSO/external writer
 - ▶ No specific data, but indicates frequency of use, who is making requests, and time to completes (delta in trace time stamps)
 - ▶ Can detect run away PSO applications
- TRACE 28 and 29 - SAPI
 - ▶ Similar to trace 27 but for the SAPI interface
 - ▶ Detects excessive SAPI requests

PERFDATA? CTRACE?

- \$D PERFDATA can help tune systems
 - ▶ QSUSE info identifies impact of CKPT delays
 - ▶ EVENT identifies commands take more than 5 seconds to complete
 - ▶ PCESTAT displays all that is known about PCE performance
- CTRACE rolling traces in dumps can help
 - ▶ Of limited use. Mostly for performance bugs.
 - ▶ Can tell how long PCE was running.

Questions?