January 2010

Performance Analysis Of JES2 Job Submission

Kris Puzza
z/OS JES2

Table of Contents

Introduction

Managing batch workload is an ongoing challenge for many customers. Several batch management products offer various solutions, but the issue usually boils down to how quickly jobs can be submitted to the job entry subsystem. Understanding the factors affecting performance of the job submission process is key to optimizing job submission rates, especially when most jobs are being submitted via a batch management product. This paper will provide an overview of the job submission process and a detailed method of analysis using JES2 PERFDATA and traces. This method will be helpful in identifying any delays and in which part of the process the delays occur.

This paper is intended for experienced system programmers with knowledge of how z/OS® and JES2 function, and some understanding of JES2 processes. Definitions of some JES2 terminology is included in the appendix. Basic knowledge of IPCS for dump analysis is also assumed.

Background

z/OS 1.7 introduced support for JES2 NJE over TCP/IP, and with that support created a new address space, the NETSERV address space. This design isolated JES2's NJE TCP/IP processing from the JES2 address space, so outages of the JES2 address space would not affect the NJE connections established by and associated with the NETSERV address space, and vice versa.

This design also made it possible to spool jobs received via NJE TCP/IP directly from the NETSERV address space. This ability to spool jobs outside of the JES2 address space allowed JES2 to redesign internal reader processing as well, exploiting this design and allowing jobs to be spooled directly from the user's (submitter's) address space. Prior to 1.7, input processing required jobs to be buffered in a dataspace, and the spooling of jobs was scheduled as work dispatched under the main task in the JES2 address space. This new design streamlined the input process, and relieved some burden from the JES2 main task. One notable difference in the design is the dispatching priority of job spooling. Prior to z/OS 1.7, job spooling was done at JES2's dispatching priority, which is typically set high due to classification in SYSSTC. With z/OS 1.7 and beyond, the dispatching priority of the user will determine the access to CPU for spooling jobs.

Overview of Internal Reader Job Submission

The process behind getting a job submitted to an internal reader involves user/application code, JES2 code running in the user/application address space, and some processing in the JES2 address space. A user/application allocates, opens, and writes to an internal reader, passing as input to each PUT a card image of a job stream. JES2 has common code in CSA/LPA to handle spooling for NETSERV as well as internal reader job submission. The PUT invokes this JES2 common code which runs on behalf of the user/application task to read in, parse, and spool each card image of the job stream, passing control to any user-defined exits throughout this process. A pseudo-job structure is built, and once the job is complete a request is queued to JES2 to instantiate the job - build the real job structure and checkpoint it.

Note that this redesigned process also shifted the bulk of the input process to the user's address space. As a result, the priority of the submitting job is a bigger consideration than prior to 1.7 in terms of how it affects the speed at which a job is submitted. Lower priority users/applications will not be able to submit jobs as fast as higher priority users/applications. Although job submission still requires some processing to occur in the JES2 address space to build key job control blocks and checkpoint the job, the overall dependency on the checkpoint cycle is significantly reduced.

Recognizing a Problem with Job Submission

If there is a perceived delay in job submission, define it. How is the delay noticed? Are any messages being used to measure the delay? If so, at what point in processing are these messages issued? Often times a delay is reported as the time between $HASP100 messages ($HASP100 job ON INTRDR) which is issued after the JOB statement is complete, but before the entire job stream has been submitted, parsed, and processed. Sometimes a delay is reported based on the time between application messages issued after each job is submitted. Is the delay consistent throughout the day/week, or is it intermittent? A delay that is noticed only at 2am each Tuesday during the heaviest batch influx may not be a delay at 3pm. Is the delay associated with a specific submitter application, or does it include any type of job submit, including TSO submit? Is the problem limited to a specific member or set of members? If multiple members, what is common? What changes have occurred recently (maintenance, upgrades, tuning, etc.)? The answers to these questions frame the problem and will determine where and when to collect documentation. For example, if there seems to be a delay with a specific submitter application on member A and it occurs only between midnight and 5am, then doc collection should be taken on member A between midnight and 5am, and also perhaps at a normal time for comparison against normal, acceptable job submission.

Planning and Collecting Documentation for Analysis

JES2 internal tracing and JES2 PERFDATA displays along with a dump are suggested documentation to begin analysis of the job submission process. The JES2 trace options are documented in the JES2 DIAGNOSIS manual. JES2 tracing provides the ability to collect extensive detailed trace events throughout JES2 processing. The trace events related to job submission will be detailed here.

JES2 also provides internal performance statistics which can be displayed with the $D PERFDATA operator command. PERFDATA provides extensive statistics on various aspects of JES2 processing: initialization, checkpoint processing, PCE processing, WLM sampling, subtask processing, and noted events. This paper will provide a limited description of PERFDATA as it pertains to analysis of job submission.

The suggested documentation combines a JES2 trace of the internal reader processing that occurs in the user/application address space in the JES2 common code, as well as a trace of the processing that occurs in the JES2 address space to create the job - the JQRP PCE. The JES2 trace will contain $SAVE and $RETURN trace entries for all JES2 processing which occurs in the user's address space and in the JES2

address space under the JQRP PCE. $SAVE and $RETURN are issued at entry and exit to each JES2 functional routine. Note that the trace below will contain entries for all user address spaces. This may be useful to compare the activity in an address space of interest to other address spaces also submitting jobs, including TSO users.

The following sequence is the suggested documentation to begin with:

1. Start trace:
   $TTRACEDEF,TABLES=20,LOG=(CLASS=H,SIZE=64000)
   $TTRACEDEF,ACTIVE=Y,LOG=(START=YES)
   $STRACE(11-12)
2. Reset PERFDATA:
   $T PERFDATA(*),RESET
3. Activate trace:
   $TRDI(*),TRACE=YES
   $T PCE(JQRP),TRACE=YES
   allow trace to run to capture job submission and
   demonstrate any suspected delays (10 min.)
4. Display PERFDATA:
   $D PERFDATA(*) (shows all PERFDATA statistics)
   or $D PERFDATA(PCESTAT) (shows only PCE statistics)
5. Dump:
   DUMP COMM=(xxx)
   x,JOBNAME=(JES2,submitter-name),
   SDATA=(NUC,SQA,CSA,RGN,TRT,SUM,GRSQ),END
   take an SVC dump of submitter and JES2 (together)
6. Stop trace:
   $PTRACE(11-12)
   $TTRACEDEF,SPIN,ACTIVE=N
   $TTRACEDEF,LOG=(START=NO)
7. Collect trace, syslog and dump

Note: Job filtering is available starting in z/OS 1.10. To trace only for a specific job or ASID, specify $TTRACE(11-12),JOBNAME= or ASID=.

The PERFDATA displays in SYSLOG will show internal performance statistics which are continuously collected by JES2, but only for processes which occur in the JES2 address space. Our interest will focus on the PCE statistics, particularly those for the JQRP PCE, as described earlier.

The SYSLOG may also be useful to identify any messages issued by the submitter, particularly in between jobs being submitted.

The dump is useful in identifying address spaces active in the trace, other processing in the submitter and JES2 address spaces, as well as providing a picture of overall system conditions. More on this later.

Analysis Approach

Analysis of the documentation collected will be detailed next - trace, PERFDATA, and dump analysis. Clues in one may determine what to look for in another, so it may be necessary to revisit documentation already reviewed.

It is also important to emphasize that although the analysis will include scrutiny of elapsed times and rates of various events and processes, it is extremely difficult to define what is “normal”. Variations in configurations, including hardware, LPAR settings, storage, system workload, and dispatching priority make each environment unique. Instead, it is more reasonable to look for anomalies within the data relative to other events or ideally, comparisons with data captured when no problem is observed.

Trace Analysis

Beginning with the trace and the dump, identify the address space of interest, the submitter (TSO user, job scheduler, etc.) to focus on. Use IPCS command IP SELECT JOBNAME(x) where x is the name of the submitter to identify the ASID where this job is executing. Next, in the trace, locate the beginning of a job being submitted by this address space by searching for 'JOB' (note the leading space) and looking for a job card image on the right. Here is an example:

Example 1:

```
16.04.35.571 ID =  11 $SAVE    ASID 01D0      009B5B08  CIRDRPUT  R14-R1 = 80D0F820 1CFDB078 00090434 7F3A2A00  --
3C3900  6161D7D8 F8F4F1F0 F2C94040 D1D6C240   4DF46BC8 F2F5F95D 6BE2F0F4 60D7C6C9  *//PQ84102I  JOB (4,H259),S04-PFI*
3C3920  60C8F2F7 F06BC3D3 C1E2E27E C46BD4E2   C7C3D3C1 E2E27ED4 6B404040 40404040  *-H270,CLASS=D,MSGCLASS=M,       *
3C3940  40404040 40403140 C3C160F7 F1404040                                      *      .                  *
```

Back up 1 line to the $SAVE entry for routine CIRDRPUT which begins this trace record. The example above shows the entry ($SAVE) in routine CIRDRPUT in ASID x'1D0', this is the first JES2 routine called after the user issues PUT for the JOB card of the job being submitted, in this case job PQ84102I. Further in the trace will be the $RETURN CIRDRPUT showing the completion of PUT processing for this JOB card. There will be a $SAVE/$RETURN pair for CIRDRPUT for each card image for the job being submitted.

Locate the end of the job by searching for ' JOB' again and finding the next JOB card being submitted and then backing up to locate the $RETURN from CIRDRPUT preceding the $SAVE for CIRDRPUT for the next job. As shown in Example 2:

Example 2:

```
16.04.36.548 ID =  12 $RETURN  ASID 01D0      009ED318  CIRDRPUT  R14-R1 = 80D0F820 00000000 00D0C9D0 7F3A2A00  --
16.04.36.556 ID =  11 $SAVE    ASID 01D0      009B5B08  CIRDRPUT  R14-R1 = 80D0F820 1CFDB078 00090434 7F3A2A00  --
3C3900  6161D7D8 F0F4F9F0 F3F94040 D1D6C240   4DF46BC8 F2F5F95D 6BE2F0F4 60D7C6C9  *//PQ049039  JOB (4,H259),S04-PFI*
3C3920  60C8F2F7 F06BC3D3 C1E2E27E C46BD4E2   C7C3D3C1 E2E27ED4 6B404040 40404040  *-H270,CLASS=D,MSGCLASS=M,       *
3C3940  40404040 40403140 C3C160F7 F1404040                                      *      . CA-71            *
```

In Examples 1 and 2 above, we have determined the following bounds of JES2 processing:

```
16.04.35.571    start of JES2 processing for PUT of jobcard for job PQ84102I
16.04.36.548    completion of JES2 processing for PUT of last card for job PQ84102I
16.04.36.556    start of JES2 processing for PUT of jobcard for NEXT job,PQ049039
```

Use this method to identify the start and end of each job being submitted. If the trace contains multiple address spaces, paying close attention to the ASID and TCB address (field after ASID value) of the trace entries to be sure they belong to the submitter of interest. Keep in mind that even within a submitting ASID, there may be multiple TCBs each submitting jobs.

Note the timestamps; the trace entries between the start and end of a job include all JES2 processing required to submit that job, but the timestamps represent the elapsed time, including interrupts and submitter processing between PUTs.

After isolating the start/end of a job submission, look closer at the trace entries in between. Note the timestamps and look for delays:

Example 3:

```
16.04.35.642 ID =  11 $SAVE    ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CFDC528 1CF7D110 7F347400 7F414960  --
16.04.35.642 ID =  12 $RETURN  ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CFDC528 00000000 00000000 7F414960  --
16.04.35.642 ID =  11 $SAVE    ASID 01D0          009ED318  JOBVALM   R14-R1 = 9CFDC110 1CF71FA0 00000000 7F3A2D0C  --
16.04.35.642 ID =  11 $SAVE    ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF725C2 1CF7D110 7F347400 7F414960  --
16.04.35.828 ID =  12 $RETURN  ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF725C2 00000000 00000000 7F414960  --
16.04.35.828 ID =  11 $SAVE    ASID 01D0          009ED318  $DESTCHK  R14-R1 = 9CF728EE 1CF99A48 00000008 21DA8BA0  --
16.04.35.828 ID =  12 $RETURN  ASID 01D0          009ED318  $DESTCHK  R14-R1 = 9CF728EE 00000000 00000008 00010000  --
16.04.35.828 ID =  12 $RETURN  ASID 01D0          009ED318  JOBVALM   R14-R1 = 9CFDC110 00000000 00000000 7F3A2D0C  --
16.04.35.828 ID =  11 $SAVE    ASID 01D0          009ED318  RPDBSEC   R14-R1 = 9CFDC2DC 1CF72E90 00000000 7F3A2D0C  --
16.04.35.828 ID =  11 $SAVE    ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FAA 1CF899D8 0000002C 7F3253D4  --
16.04.35.828 ID =  12 $RETURN  ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FAA 00000000 0000002C 0000002B  --
16.04.35.828 ID =  11 $SAVE    ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FD8 1CF899D8 00000035 7F414C6C  --
16.04.35.828 ID =  12 $RETURN  ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FD8 00000000 00000035 00000030  --
16.04.35.828 ID =  11 $SAVE    ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF7306E 1CF7D110 7F347400 7F414960  --
16.04.35.926 ID =  12 $RETURN  ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF7306E 00000000 00000000 7F414960  --
16.04.35.926 ID =  11 $SAVE    ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FAA 1CF899D8 0000002C 7F325554  --
16.04.35.926 ID =  12 $RETURN  ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FAA 00000000 0000002C 0000002B  --
16.04.35.926 ID =  11 $SAVE    ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FD8 1CF899D8 00000035 7F414C6C  --
16.04.35.926 ID =  12 $RETURN  ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FD8 00000000 00000035 00000030  --
16.04.35.926 ID =  11 $SAVE    ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF7306E 1CF7D110 7F347400 7F414960  --
16.04.36.044 ID =  12 $RETURN  ASID 01D0          009ED318  $RACROUT  R14-R1 = 9CF7306E 00000000 00000000 7F414960  --
16.04.36.044 ID =  11 $SAVE    ASID 01D0          009ED318  DSNCMP    R14-R1 = 9CF72FAA 1CF899D8 0000002C 7F3256D4  --
```

In Example 3 above, delays are noted across $RACROUTE calls. In this particular case, further analysis determined that an improper restructure of the security database caused delays in certain types of security requests.

The trace in Example 4 below shows the transition from processing in the user/application address space to the JES2 address space's JQRP PCE via a $JQESERV request, and the completion of the $JQESERV request as the user/application is posted and resumes. This sequence shows the normal sequence of a $JQESERV request issued in the submitter address space ASID 1D0, followed by the JQRP PCE in the JES2 address space creating the new job (JOB00072) before posting the submitter back for completion of the $JQESERV request:

Example 4:

```
16.04.51.875 ID =  11 $SAVE    ASID 01D0           009B5B08  $JQESERV  R14-R1 = 9CFDBC5A 7F30E000 00000000 7F4176B4  --
16.04.51.875 ID =  11 $SAVE    ASID 01D0           009B5B08  $IOTBLD   R14-R1 = 9CF6FF4C 00000000 7F2FE000 7F2FE000  --
16.04.51.875 ID =  12 $RETURN  ASID 01D0           009B5B08  $IOTBLD   R14-R1 = 9CF6FF4C 00000000 7F2FE000 7F2FE000  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  GETJOBKY  R14-R1 = 9E07EEEE 000121B0 1E5C1CC8 00000000  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  GETJOBKY  R14-R1 = 9E07EEEE 00000000 1E5C1CC8 00000000  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $QADD     R14-R1 = 9E07EF28 1E074130 00000020 1E5C1B1C  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0742B6 9E0742A8 00000000 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A22A 1E0A64C8 00000000 2068E198  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A22A 00000000 00000000 2068E198  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0742B6 00000000 1E5C136C 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E074304 9E0742F6 1E5C136C 00000000  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A642 1E0A64C8 00000000 2068E198  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A642 00000000 064586FF 2068E198  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E074304 00000000 00000000 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E074F6C 9E074F5E 1F32F6E8 00010001  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E074F6C 00000004 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0763F6 9E0763E8 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0763F6 00000004 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $QACT     R14-R1 = 9E07433E 1E077400 00000000 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0774EC 9E0774DE 1F32F6E8 1F1C42D6  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E0774EC 00000004 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $QACT     R14-R1 = 9E07433E 00000000 00000000 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN            JQRP      1E1CD668  $QADD     R14-R1 = 9E07EF2A 00000000 00000020 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE              JQRP      1E1CD668  $QJIX     R14-R1 = 9E07EF56 1E077508 0011C6D0 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E077BEA 9E077BDC 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E077BEA 00000004 1F32F6E8 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $QJIX     R14-R1 = 9E07EF5A 00000000 0011C6D0 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E07EFE2 9E07EFD4 1E5C1B1C 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A22A 1E0A64C8 00000048 2068E198  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $DOGBERT  R14-R1 = 9E07A22A 00000000 00000000 2068E198  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $DOGJQE   R14-R1 = 9E07EFE2 00000000 1E5C1B1C 1F32F6E8  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $GETBUF   R14-R1 = 9E07F038 0000CBF0 1E5C1B1C 1F32F6E8  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $GETBUF   R14-R1 = 9E07F038 00000000 1E5C1B1C 1E2DA000  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $GETBUF   R14-R1 = 9E07F0B0 0000CBF0 1E2DB000 00000000  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $GETBUF   R14-R1 = 9E07F0B0 00000000 1E2DB000 1E2DB000  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $TRACK    R14-R1 = 9E07F160 1E0C8040 1F32F6E8 1E2DB0D8  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  TBLOBTGB  R14-R1 = 9E0C82F2 00000000 1F32F6E8 00000000  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  TBLOBTGB  R14-R1 = 9E0C82F2 00000004 1F32F6E8 00000000  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  TBLOBTGB  R14-R1 = 9E0C82F2 00000000 1F32F6E8 00000000  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  SIGIO     R14-R1 = 9E0C8B82 1E16E0AC 00000001 1E16E0B4  --
16.04.52.672 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  SIGIO     R14-R1 = 9E0C8B82 00000000 00000001 1E16E0B4  --
16.04.52.672 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  SIGIO     R14-R1 = 9E0C8D2A 1E16E0AC 00000001 1E16E0B4  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  SIGIO     R14-R1 = 9E0C8D2A 00000000 00000001 1E16E0B4  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  TBLOBTGB  R14-R1 = 9E0C82F2 00000000 1F32F6E8 00000000  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $TRACK    R14-R1 = 9E07F160 00000004 1F32F6E8 14575301  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  JQRJCTA   R14-R1 = 9E0A55B0 1E07F884 14575301 1F32F6E8  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  JQRJCTA   R14-R1 = 9E0A55B0 00000000 14575301 1F32F6E8  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $TRACK    R14-R1 = 9E07F1A4 1E0C8040 1F32F6E8 1E2DB0D8  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $TRACK    R14-R1 = 9E07F1A4 00000000 1F32F6E8 14575302  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $CBIOM    R14-R1 = 9E07F1DC 1F32F6E8 1E2DB078 00000000  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  CBMWRTN   R14-R1 = 8000EFA2 0000F302 1E2DB078 1E2DB000  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $EXCP     R14-R1 = 8000F540 0000A248 1E2DB000 1E1CD74C  --
 1E1CD74C  01040003 1E1CD668 00000000 0013F700  14575302 1E1CD668 00000100           *.....O.......7.......O.....    *
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $EXCP     R14-R1 = 8000F542 00000000 1E2DB000 1E1CD74C  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  CBMWRTN   R14-R1 = 8000EFA2 00000000 1E2DB078 1E2DB000  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $CBIOM    R14-R1 = 9E07F1DC 00000000 1E2DB078 1E2DB000  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $CBIOM    R14-R1 = 9E07F214 1F32F6E8 1E2DA078 00000000  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  CBMWRTN   R14-R1 = 8000EFA2 0000F302 1E2DA078 1E2DA000  --
16.04.52.684 ID =  11 $SAVE    JOB00072  JQRP      1E1CD668  $EXCP     R14-R1 = 8000F540 0000A248 1E2DA000 1E1CD74C  --
 1E1CD74C  01040003 1E1CD668 00000000 0013F620  14575301 1E1CD668 00010100           *.....O.......6.......O.....    *
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $EXCP     R14-R1 = 8000F542 00000000 1E2DA000 1E1CD74C  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  CBMWRTN   R14-R1 = 8000EFA2 00000000 1E2DA078 1E2DA000  --
16.04.52.684 ID =  12 $RETURN  JOB00072  JQRP      1E1CD668  $CBIOM    R14-R1 = 9E07F214 00000000 1E2DA078 1E2DA000  --
16.04.52.687 ID =  12 $RETURN  ASID 01D0           009B5B08  $JQESERV  R14-R1 = 9CFDBC5A 00000000 7F30E000 7F4176B4  --
```

Next, collect a series of start/end timestamps for job submits to get a larger picture of elapsed times. The following pairs are start/end timestamps from CIRDRPUT $SAVE for the job card to CIRDRPUT $RETURN for the last statement before the next job card, as shown above. (These start/end timestamps are manually collected by navigating the trace as described with Examples 1 and 2 above.) Also, collect a sampling of such sequences throughout the duration of the trace, as indicated below by skipping intervals of time:

Example 5:

```
00.00.58.294\ start
00.00.58.754/ end

00.00.58.783\
00.00.59.027/

00.00.59.069\
00.00.59.484/

00.00.59.517\
00.00.59.932/

00.00.59.965\
00.01.00.247/

skip 2 minutes of trace data

00.02.57.260\
00.02.57.436/

00.02.57.455\
00.02.57.902/


00.02.57.941\
00.02.57.985/

00.02.58.005\
00.02.58.226/

00.02.58.272\
00.02.59.291/

00.02.59.311\
00.02.59.729/

00.02.59.748\
00.02.59.966/

00.02.59.987\
00.03.00.438/

00.03.00.456\
00.03.00.882/

skip 3 minutes of trace data
```

```
00.05.57.952\
00.05.58.552/

00.05.58.572\
00.05.58.942/

00.05.58.957\
00.05.59.386/

00.05.59.399\
00.05.59.734/

00.05.59.752\
00.06.00.049/

00.06.00.070\
00.06.00.524/

00.06.00.540\
00.06.00.868/

skip 4 minutes of trace data

00.09.58.096\
00.09.58.398/

00.09.58.450\
00.09.58.838/

00.09.58.851\
00.09.59.187/

00.09.59.209\
00.09.59.528/

00.09.59.544\
00.09.59.632/

00.09.59.689\
00.10.00.139/
```

Each pair above represents total elapsed time for JES2 processing - parsing, spooling, security, and exits in the user/application address space, and JQRP PCE processing in the JES2 address space. Example 5 above shows reasonable elapsed times for this particular installation, and the samples taken throughout the trace also shows the times are consistent, there are no significant variations in the times.

Also note the time intervals from the end of one job submit to the start of the next job submit. This is the elapsed time spent between JES2 requests and can indicate other processing occurring in the submitter job/application. A large elapsed time here implies delays in the application between job submits, as shown in Example 6 below:

Example 6:

```
18.50.26.96948\ JES2 start
18.50.27.02461/ JES2 end                      approx. 0.05 seconds

   submitter time before issuing next PUT - approx. 9.41 seconds

18.50.36.43302\ JES2 start
18.50.37.53781/ JES2 end                      approx. 1.10 seconds

   submitter time before issuing next PUT - approx. 9.10 seconds

18.50.46.63230\ JES2 start
18.50.46.69462/ JES2 end                      approx. 0.06 seconds
```

This could indicate delays in the submitter application (waits, interrupts). Examination of the system trace table in the SVCDUMP filtered for the submitter's address space may show what processing occurs in the application between PUTs. Additional analysis may require more specific traces in the application to understand these delays.

PERFDATA Analysis

The JES2 PERFDATA statistics are collected continuously by JES2 while it runs, gathering information about various processes it performs. The statistics are only collected for processes that occur in the JES2 address space, no statistics are collected for JES2 functions that run in the user environment. The statistics are accumulated beginning when JES2 starts and are reset only via a $T PERFDATA, RESET command. PERFDATA is displayed via $HASP660 in response to the $D PERFDATA command.

When collecting PERFDATA for analysis, it is important to first issue $T PERFDATA, RESET to clear the statistics and begin accumulating new statistics for the time frame of interest, this avoids ambiguous statistics of a potentially very long interval, possibly since JES2 started, where any data of interest would be lost or "watered down". The technique used in the suggested documentation described earlier resets PERFDATA, and then displays PERFDATA after a 10 minute interval. Another technique is to automate the resets and displays on some regular interval.

Once the desired PERFDATA displays have been done, SYSLOG (or OPERLOG) is collected for analysis. To examine the JES2 address space's role in the job submission process, analysis will focus on the JQRP PCE statistics. Begin by searching for "PCE PERFORMACE STATISTICS" to locate the beginning of all PCE statistics:

Example 7:

```
PCE PERFORMANCE STATISTICS - INTERVAL=6:05.581740,
CPU=13.170489,
PCENAME=ASYNC,TIME=0.591494,CPU=0.268106,CPU%=0.50,
QSUSE_TIME=0.000000,IOCOUNT=0,CKPT_COUNT=0,
  WAIT=WORK,MOD=HASPNUC,SEQ=64570000
   COUNT=15384,AVGWAIT=0.047747,
     POST=$$POST,COUNT=15384,AVGWAIT=0.047747,
```

Note the interval, this is the wall-clock elapsed time between the last reset and the display. CPU time is total CPU time for all PCEs. A very large disparity between these times (large interval, very little CPU time) would indicate that JES2 is not getting much CPU and could be a dispatching priority issue. Example 7 above is reasonable; an interval of 6 minutes and CPU time less than a second would be a problem warranting further investigation of JES2 dispatching priority or any unusual activity in other higher priority work. More on this later when dump analysis is discussed.

Assuming that the interval/CPU times indicate JES2 does not appear starved for CPU, locate the JQRP PCE within the PCE PERFORMANCE STATISTICS by searching on "PCENAME=" and stepping through the PCE sections until the JQRP PCE statistics are located:

Example 8:

```
$HASP660 PCENAME=JQRP,TIME=0.903077,CPU=0.700405,CPU%=3.13,                $HASP660
QSUSE_TIME=0.278596,IOCOUNT=2196,CKPT_COUNT=35317,                         $HASP660
WAIT=WORK,INHIBIT=NO,MOD=HASPJQS,SEQ=70910000                              $HASP660
COUNT=2061,AVGWAIT=0.329332,                                               $HASP660
POST=IO,COUNT=936,AVGWAIT=0.000606,                                        $HASP660
POST=$$POST,COUNT=1125,AVGWAIT=0.602831,                                   $HASP660
WAIT=CKPT,INHIBIT=NO,MOD=HASPJQS,SEQ=70923300                              $HASP660
COUNT=3542,AVGWAIT=0.095666,                                               $HASP660
POST=RESOURCE,COUNT=1528,AVGWAIT=0.188931,                                 $HASP660
POST=IO,COUNT=947,AVGWAIT=0.000848,
```

The PERFDATA PCE statistics for the JQRP PCE shows the I/O count for the interval.

Since JES2 performs 2 I/Os for each job created, that number divided by 2 will give a count of how many jobs were created during that interval. In Example 8 above, 2196/2=1098 jobs in the 6:05.581740 interval, or approximately 3.0 jobs per second. Calculating this rate is helpful when comparing PERFDATA reports for different times, intervals, and systems. Remember, these statistics represent ALL jobs created for all jobs input on this member, not just for one internal reader submitter.

The obvious next question is - what is a good rate? As stated earlier, that question is difficult to answer and varies from customer to customer because of workload, configuration, LPAR factors, etc. A more important question is, what is the rate when behavior is normal, and how does this compare to the rate when behavior is not normal?

Also note the average checkpoint wait time:

Example 9:

```
$HASP660 PCENAME=JQRP,TIME=0.903077,CPU=0.700405,CPU%=3.13,                $HASP660
QSUSE_TIME=0.278596,IOCOUNT=2196,CKPT_COUNT=35317,                         $HASP660
WAIT=WORK,INHIBIT=NO,MOD=HASPJQS,SEQ=70910000                              $HASP660
COUNT=2061,AVGWAIT=0.329332,                                               $HASP660
POST=IO,COUNT=936,AVGWAIT=0.000606,                                        $HASP660
POST=$$POST,COUNT=1125,AVGWAIT=0.602831,                                   $HASP660
WAIT=CKPT,INHIBIT=NO,MOD=HASPJQS,SEQ=70923300                              $HASP660
```

```
COUNT=3542,AVGWAIT=0.095666,                                    $HASP660
POST=RESOURCE,COUNT=1528,AVGWAIT=0.188931,                      $HASP660
POST=IO,COUNT=947,AVGWAIT=0.000848,
```

This represents the average time the PCE waited for access to the JES2 checkpoint. Look for large average checkpoint wait time to indicate contention for checkpoint, which will be examined next.

Checkpoint Considerations

If average wait for checkpoint under the JQRP PCE indicates a possibility of contention for checkpoint, a MAS-wide view must be considered:

1) JES2 parameters MASDEF HOLD and DORMANCY determine how long a JES2 member will hold the checkpoint reserve and how long it will refrain from attempting to reserve the checkpoint again after releasing it. Have there been any recent changes to HOLD/DORMANCY on any members recently? If so, for what purpose?
2) Has there been a change in workload anywhere in the MAS? A sudden influx of jobs being submitted? An increase in some SAPI application processing (archivers, print applications, etc.) which is demanding more access to checkpoint?
3) Are there any JES2 commands (possibly automated) that require extensive job queue processing (and therefore checkpoint access) at the time of the delays?
4) Is there some activity elsewhere in the MAS that is monopolizing the checkpoint and starving the member where job submission is being examined?

PERFDATA displays from all members can be further examined for details on checkpoint performance. In JES2 PERFDATA, the CKPTSTAT checkpoint performance statistics provide the average HOLD and DORMANCY values for the given interval, as well as the total number of $CKPTs on the member for the given interval:

Example 10:

```
$HASP660 CKPT PERFORMANCE STATISTICS - INTERVAL=11:10:12.320961,
$HASP660 AVGHOLD=0.318337,AVGDORM=45.305289,TOT$CKPT=3284,
$HASP660 WRITE-4K=0,WRITE-CB=788,OPT$CKPT=2496,OPT4K=0,
$HASP660 IO=R1,COUNT=875,AVGTIME=0.010943,
$HASP660 IO=R2,COUNT=0,AVGTIME=0.000000,TOTAL4K=0,TOTALCB=118,
$HASP660 IO=PW,COUNT=876,AVGTIME=0.004066,TOTAL4K=39,TOTALCB=0,
$HASP660 IO=IW,COUNT=878,AVGTIME=0.003776,TOTAL4K=0,TOTALCB=670,
$HASP660 IO=FW,COUNT=876,AVGTIME=0.003888,TOTAL4K=0,TOTALCB=118
```

A comparison of these PERFDATA statistics across the MAS will show which members have the most checkpoint activity (TOT$CKPT) and, on average, how long each member is holding and waiting for the checkpoint. In addition, JES2 TRACE ID 17 gives detailed records showing the actual HOLD and DORMANCY values for each checkpoint cycle for more detailed analysis.

See the JES2 Diagnosis manual for further information about TRACE ID 17.

Again, it is difficult to identify "normal" average HOLD and DORMANCY values without understanding the MASDEF settings on each member and the rationale used to establish these settings. Generally, compare average values from PERFDATA (or actual values from TRACE ID 17) against MASDEF settings and look for disparity.

Also note the average I/O times for checkpoint (green in Example 10 above). These I/O times are for CKPT1 and CKPT2, and are measured from EXCP to completion, including the POST for I/O completion and dispatch of the JES2 main task. Note these times cannot be compared to RMF™ I/O statistics because they are measured differently. Depending on checkpoint placement (coupling facility, DASD, both) these average I/O times can vary. High I/O times can occur when DASD checkpoint volumes are mirrored via XRC and an improper pacing value is defined.

If PERFDATA analysis of members across the MAS indicates contention for checkpoint and a member is found to be monopolizing the checkpoint (large average HOLD, lowest average checkpoint wait, high CKPT_COUNT) , further analysis of PERFDATA on that member can identify which process is causing the demand on checkpoint. JES2 PERFDATA CPUSTAT cpu statistics shows CPU utilization by JES2 process:

Example 11:

```
CPU PERFORMANCE STATISTICS - INTERVAL=9:36.179855,CPU=
49.335859,
PCENAME=CKPT,CPU%=58.14,CPU=38.555304,TIME=40.736316,
QSUSE_TIME=0.572341,IOCOUNT=189,CKPT_COUNT=3795,
PCENAME=SPI,CPU%=25.94,CPU=2.935075,TIME=3.077212,
QSUSE_TIME=3.059467,IOCOUNT=6208,CKPT_COUNT=19084,
PCENAME=HOPE,CPU%=4.69,CPU=2.318661,TIME=2.448082,
QSUSE_TIME=2.337202,IOCOUNT=1287,CKPT_COUNT=5401,
PCENAME=XCFCMND,CPU%=3.31,CPU=1.634068,TIME=1.680183,
QSUSE_TIME=0.000000,IOCOUNT=0,CKPT_COUNT=0,
PCENAME=COMM,CPU%=2.37,CPU=1.172884,TIME=1.248076,
QSUSE_TIME=1.220225,IOCOUNT=62,CKPT_COUNT=3307,
PCENAME=SPOOL,CPU%=1.40,CPU=0.693801,TIME=0.733589,
QSUSE_TIME=0.729423,IOCOUNT=0,CKPT_COUNT=201584,
PCENAME=PSO,CPU%=1.07,CPU=0.531341,TIME=0.559536,
QSUSE_TIME=0.532962,IOCOUNT=208,CKPT_COUNT=353,
PCENAME=SPIN,CPU%=0.83,CPU=0.410516,TIME=0.429190,
QSUSE_TIME=0.427724,IOCOUNT=60,CKPT_COUNT=785,
PCENAME=EXEC,CPU%=0.61,CPU=0.302569,TIME=0.404187,
QSUSE_TIME=0.337195,IOCOUNT=0,CKPT_COUNT=5360,
PCENAME=PURGE,CPU%=0.52,CPU=0.257058,TIME=0.273014,
QSUSE_TIME=0.151292,IOCOUNT=1656,CKPT_COUNT=2322,
PCENAME=TIMER,CPU%=0.10,CPU=0.054258,TIME=0.056381,
QSUSE_TIME=0.000000,IOCOUNT=0,CKPT_COUNT=0,
PCENAME=SNF,CPU%=0.05,CPU=0.026058,TIME=0.028013,
QSUSE_TIME=0.001161,IOCOUNT=0,CKPT_COUNT=0,
PCENAME=CNVT,CPU%=0.04,CPU=0.019985,TIME=0.021590,
QSUSE_TIME=0.020459,IOCOUNT=50,CKPT_COUNT=352,
```

```
PCENAME=MCON,CPU%=0.03,CPU=0.016689,TIME=0.018406,
QSUSE_TIME=0.002379,IOCOUNT=21,CKPT_COUNT=63,
PCENAME=RESOURCE,CPU%=0.02,CPU=0.010440,TIME=0.010770,
QSUSE_TIME=0.000000,IOCOUNT=0,CKPT_COUNT=0,
PCENAME=JQRP,CPU%=0.01,CPU=0.006006,TIME=0.006170,
QSUSE_TIME=0.001909,IOCOUNT=18,CKPT_COUNT=271,
```

In Example 11 above, the SAPI process (PCENAME=SPI) is very busy and has a relatively high checkpoint count. This suggests that the SAPI applications on this member should be investigated, possibly with further JES2 SAPI traces in place. (Note that for CPU statistics, JES2's checkpoint PCE is normally the busiest of all processes.)

Tuning the JES2 Checkpoint MASDEF HOLD and DORMANCY values to balance the needs of all demands across the MAS - job submits, TSO logons, printers, SAPI applications, NJE, batch, etc. - is not a simple task, it is unique to each customer environment, and is well beyond the scope of this paper.

Dump Analysis

Analysis of the SVC dump taken during any observed delays can provide additional details about the submitter and JES2, as well as overall system conditions. The system trace table gives a snapshot of system activity. IPCS SYSTRACE JOBNAME(xxxx) , where xxxx is the submitter, will show the activity of the submitter including possible activity between PUTs to the internal reader and between submitting jobs. If there are WAIT entries in the trace table on regular, non-zIIP/zAAP processors, all dispatchable work is being exhausted and CPUs are going idle, so dispatching priority of the submitter is not an issue. If on the other hand there are no WAIT entries, at least for the duration of the trace table the system is busy and dispatching priority may be a factor. In that case, review the trace table to determine which higher priority work is running, and at what dispatching priority.

IPCS SYSTRACE JOBNAME(JES2) filters the system trace table for only JES2 address space activity. Since JES2 runs mostly enabled, the system trace table provides a good picture of JES2 activity. If PERFDATA analysis indicated that overall JES2 CPU is high or that a particular

PCE has high CPU, the trace table filtered for JES2 may show a repetitive process; look for I/O and EXT interrupts with similar PSW addresses. Identify the code where the PSWs are executing, this could be JES2 code, user exits, or other system code.

IPCS SYSTRACE ALL displays the full system trace table with all activity for all address spaces. If PERFDATA analysis indicated that JES2 is starved for CPU, examine the system trace table to identify what IS running, and at what dispatching priority compared to JES2.

IP SYSTRACE JOBNAME(JES2,xxxx) filters the trace table for only JES2 and the submitter, and will show interaction of the submitter and JES2 address spaces. Activity in the JES2 address space may identify intensive processing identified by I/O, EXT, CLKC interrupts with the PSW addresses in a specific area. In

this case, identify the module pointed to by the PSW, and if the activity is under the JES2 main task (PRB name HASJES20) then compare the time stamps against the JES2 dispatcher CTRACE entries to identify the active PCE. (IP CTRACE COMP (SYSJES2) SUB((DISP)) FULL)

The $JQESERV requests are queued as JQRB control blocks in the JES2JQRB dataspace anchored in CCTJQRBH. This queue represents all requests in the system, a long queue may indicate JES2 delays in processing the requests, or many more requests from other submitters, possibly at higher priority. If JES2 processing seems to be contributing to a delay, the IBM Support Center should do further analysis.

Other Considerations

Other tools may provide additional insight. RMF reports may be of interest, particularly the Device Activity report and Coupling Facility report for potential hardware delays on the checkpoint devices.

SMF type 30 records can also be used to get start and end timestamps of when jobs are recognized by JES2 internal reader processing.

JES2 monitor reports $JDSTATUS and $JDDETAILS may surface any monitor alerts including problems with the JES2 main task, or resource shortages. EREP reports may also provide hints of errors contributing to the problem.

As the source of a delay becomes suspect, additional documentation, traces, dumps, etc. may be necessary to fully identify the cause.

Summary of Analysis

After defining the perceived delay, and collection of timely documentation, the analysis is summarized as:

1) Identify the start/end times of jobs being submitted for a given submitter
2) Collect a series of job submit start/end times, note delays within and between submits
3) Establish a submission rate (x seconds per job, y jobs per second) for the submitter
4) Analyze PERFDATA for overall JES2 performance, examine JQRP PCE statistics and calculate JES2 job creation rate (x jobs per interval for ALL submits). Review checkpoint performance statistics for contention.
5) Dump analysis for activity of submitter and JES2, and overall system conditions
6) Checkpoint activity comparison MAS-wide, look for other applications/processes monopolizing checkpoint

Kris Puzza is a z/OS Software Support technical specialist and JES2 Service team leader. She has 28 years of experience in z/OS Software Support, 15 years of which are in JES2.

Acknowledgements

Many thanks to Lorne Parks for his contributions to the PERFDATA analysis.

Appendix

Publications:

JES2 Diagnosis	GA22-7531-08
JES2 Initialization and Tuning Reference	SA22-7533-08
JES2 Commands	SA22-7526-09
JES2 Messages	SA22-7537-08

Additional PERFDATA documentation:

http://www-03.**ibm.com**/support/techdocs/atsmastr.nsf/5cb5ed706d254a8186256c71006d2e0a/59ebc9c955fc619e85256d8c000d6156/$FILE/perfdataFlashzOSR2.pdf

JES2 Terminology Defined:

NJE - Network Job Entry

JQE - Job Queue Element; the main control block that represents a job, resides on the JES2 checkpoint dataset.

PCE - Processor Control Element; a sub-dispatched process that runs in the JES2 address space under the JES2 main task, each PCE has a specific purpose.

JQRP PCE - JQE Request Processor PCE - JES2 process responsible for handling requests queued to build the major job control blocks.

$JQESERV - macro to request various JQE services from the user environment, including a Request for JES2 to create a job.

SAPI - SYSOUT Application Program Interface - An SSI interface to allow applications to request access to SYSOUT datasets on spool.

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
01/2010

ZSW03142-USEN-00