MP1B: WebSphere MQ for z/OS V8.0

Displaying WebSphere MQ statistics and accounting

Colin Paice
July 2014
Document Number MP1BV80

Take Note!
Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

V800 Edition, July 2014. Printed 04/08/2014
This edition applies to Version 8.0, 7.1 and V7.0.1 of "WebSphere MQ for z/OS" - Interpreting accounting and statistics data" and to all subsequent releases and modifications until otherwise indicated in new editions.

Sending your comments to IBM
You can send your comments electronically to idrcf@uk.ibm.com.
When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you. You may continue to use the information that you supply.

Notices

This report is intended to give guidance on the use and interpretation of the statistics and accounting in WebSphere MQ for z/OS Version 8.0,7.1 and V7.0.1. The information in this report is not intended as the specification of any programming interfaces that are provided by z/OS or WebSphere MQ.

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed "as is". The use of this information, and the implementation of any of the techniques, is the responsibility of the customer, and depends on the customer's ability to evaluate and integrate them into their operational environment.

The following terms, used in this document, are trademarks of the IBM Corporation in the United States or other countries or both:

CICS

WebSphere MQ

z/OS

Table of Contents

Changes since V7

This SupportPac supports WebSphere MQ V8 and earlier releases.

V7.1 APAR PM61284: Addition to log manager statistics. This adds information about log I/O times and the longest I/O time in the SMF interval. It also adds the duration of the SMF record.

The changes include

1. Support of the SMF data for the chinit statistics and channel accounting
    1. Adapter, Dispatcher, SSL, DNS TCB information
    2. Information similar to display channel status for channels.
    3. Report if the average SSL duration is larger than a user specified value
    4. Report if the nettime is larger than a user specified value
2. Update buffer pool statistics to include 64 bit buffer pools and 'fixed' buffer pools
3. Update to log statistics showing average times for I/O requests. See APAR PM61284.
    1. Using the information allows the *log busy%* to be calculated.
4. SMF statistics have the start time and duration of the records, so rates can be calculated, such as MB logged per second per log. See APAR PM61284.
5. Improved selection criteria
    1. Display queue records only if the curdepth is greater than user specified value.
    2. Specify Jobname, such as MQ02CHIN to display records only from that job.

What you can do with the program

You can use the MQSMF program in different ways depending on your requirements.

What work is using my queue manager?
- List a summary of the transactions, jobs and channels on page 7
- List a summary of the queues being used by which application on page 7
- List the high use queues on page 7.

How do I monitor my queue manager? on page 7.

Investigate potential problems, on page 7.

Are the application well behaved? on page 8.

Is the queue manager set up OK? on page 8.

Some of the output files are in Comma Separated Value (CSV) format which can be imported into a spread sheet. The spread sheet can then be used to draw graph and display trends over time.

Summary of the transactions, jobs and channels

To get a summary of the applications and channels using MQ use the TASKCSV file, see page 73. This summarises the Accounting class 3 records by Date, Hour and work type. This report has information on the CPU used, MB of data logged, how many MB were put and got. Data for multiple instances of a transaction or a job are summarised into one record.

Summary of the queues being used by which application

To get a summary of the queues used, use the QSUML file on page 39 for information about local queues , and QSUMS file on page 39 for information about shared queues. These reports have information on the number of MB put and got, the number of valid gets and puts, the maximum queue depth and, for local queue the number of messages read from the page set

What are the high use queues?

You can use the QSUML report on page 39 and the QSUMS report on page 39, and specify parameters QueuePutMB and/or QueueGetMB to the MQSMF program. Queues which put or get less than these values are not displayed. This eliminates low usage queues, such as using a temporary dynamic queue to use the command server.

How do I monitor my queue manager

You can monitor the amount of data logged by using the log data statistics in CSV format, see page 45.
You can monitor buffer pool usage using the buffer pool statistics in CSV format, see page 50.

You should collect data for a good day, so you have a base line to compare other days with.

Investigate potential problems

In ddname //TASKSUM is a summary of messages produced when looking at the task and queue

records.
This has data like

```
Record#    Count    Value Message
   2202       25    98908 MQTASK13E long commit time C,'CP15','IYFFC000',
     38        1   106347 MQTASK13E long commit time B,'PAICEP7A',' ',
```

This has the following meaning.

- There was a message `MQTASK13E long commit time C,'CP15','IYFFC000',`
  - It was produced 25 times
  - The longest value (of the commit time) was 98908 microseconds. This was at record 2202 in the input file
- There was a message `MQTASK13E long commit time B,'PAICEP7A',' ',`
  - It was produced 1 time,
  - the largest (only) value was 106347 in record 38 of the input file.

To investigate these in more detail you can use specify parameters to the MQSMF program: StartRecord=2202, LastRecord=2202 and Detail(20). This will give all maximum level of detail for the one record.

Messages are written to the ddname //MESSAGE report to indicate possible problems, such as messages read from a page set, long CF response time. By increasing the value specified in the Detail, you can get more potential messages produced.
Messages from the CHINIT statistics are written to the ddname //CMESSAGE.

For example

MQQPST04E MVCA MQQ2 2013/02/01 12:54:27 VRM:701 BP 10 Many (11317) pages read from disk. This is typical of long lived messages. Buffer pool may be too small
MQQPST02S MVCA MQQ2 2013/02/01 12:50:46 VRM:701 BP 10 Filled many(165) times. This is typical of long lived messages. Buffer pool may be too small

Once you have identified a potential problem you can use the parameter FirstRecord and LastRecord to display a subset of the SMF data, and use Detail to display more information for this subset of records.

If you think you have problems with puts taking a long time, you can specify the Long_Put parameter, and for puts taking over this time it will display a message like
MQTASK08E Long Put time due to logging MYQUEUE

Are the application well behaved

Messages are written to the MESSAGE report when possible unusual application programming behaviour is detected. For example

- Many gets for a specific message, but the queue is not indexed.
- An application repeatedly failed to get a message from a queue. This might be caused by the common programming error, where the message-id and correlation-id are not cleared before doing a get.

Is the queue manager set up OK?

If a buffers pool is too small and frequently fills up, messages are reported in the MESSAGE report

suggesting area you may need to investigate.

How to run the program

The MQSMF program produces the output in separate files, so in the file for log statistics, you only get the log statistics.

You can use optional parameters to select which records are processed, and how much information is displayed.

The program also has 'rules' built in which displays potential out-of-line conditions. For example if the buffer pool statistics show that some gets required data to be read from the page set it will produce a message. For example.

MQQPST05I MVCA MQ7A 2013/01/06 08:00:00 VRM:710 BP 2 Some (1000) pages read from disk. Buffer pool may be too small

So using this you just need to review the messages, and not the detailed buffer pool statistics.

SMF data

SMF data can be written to data sets or to log streams in the coupling facility.
The D SMF,O operator command gives you information about your SMF set-up, for example
I*EE967I 08.42.01 SMF PARAMETERS*
RECORDING(LOGSTREAM) -- REPLY
LSNAME(IFASMF.MQ,TYPE(115,116)) -- PARMLIB
LSNAME(IFASMF.CICS,TYPE(110)) -- PARMLIB
DEFAULTLSNAME(IFASMF.DEFAULT) -- PARMLIB
DSNAME(SYS1.MVCA.MANB) -- PARMLIB
DSNAME(SYS1.MVCA.MANA) -- PARMLIB

Where
RECORDING(LOGSTREAM) shows the Coupling Facility log stream is being used.
LSNAME(IFASMF.MQ,TYPE(115,116)) shows the log stream to be used for the MQSMF records.

RECORDING(DATASET) is the default, and may not be listed.
DSNAME(SYS1.MVCA.MANA) and *DSNAME(SYS1.MVCA.MANB)* show the datasets to be used when RECORDING(DATASET) is used.

JCL to extract SMF data from log stream

```
//SMFDUMP   EXEC  PGM=IFASMFDL
//DUMPOUT  DD   DSN=&TEMP,DISP=(NEW,PASS),
//       SPACE=(CYL,(100,100),RLSE)
//SYSPRINT DD         SYSOUT=*
//SYSIN  DD *
  DATE(2013004,2013012)
  START(0900)
  END(1900)
```

```
 LSNAME(IFASMF.MQ,OPTIONS(ALL))
 OUTDD(DUMPOUT,TYPE(115,116))
/*
```

This JCL

- selects the records from the IFASMF.MQ logstream (configured to contain the MQ SMF records 115 and 116) see above
- starting from date 2013 day number 004 to 2013 day number 12
- selecting records starting from 0900
- up to and including records with a time of 1900

They get written to the dataset with ddname //DUMPOUT, which is a temporary dataset.
When running the MQSMF program, the SMFIN dataset can refer to the DUMPOUT dataset
`DSN=*.SMFDUMP.DUMPOUT.`
where

- * means from this this job
- SMFDUMP is the name of the job step that created the dataset
- DUMPOUT is the dataset within the job step that holds the SMF records.

JCL to extract SMF data from data sets

```
//SMFDUMP   EXEC  PGM=IFASMFDP
//INDD1     DD    DISP=SHR,DSN=SYS1.MVCA.MANA
//INDD2     DD    DISP=SHR,DSN=SYS1.MVCA.MANB
//DUMPOUT    DD    DSN=&TEMP,SPACE=(CYL,(10,10)),DISP=(NEW,PASS)
//SYSPRINT  DD    SYSOUT=H
//SYSIN     DD    *
INDD(INDD1,OPTIONS(DUMP))
INDD(INDD2,OPTIONS(DUMP))
OUTDD(DUMPOUT,TYPE(115,116))
START(0000)
 END(1700)
/*
```

This JCL

- selects the records from the datasets SYS1.MVCA.MANA and SYS1.MVCA.MANB
- selecting records starting from 0000
- up to and including records with a time of 1700
- they get written to the dataset with ddname //DUMPOUT, which is a temporary dataset.

When running the MQSMF program, the SMFIN dataset refers to the DUMPOUT dataset
`DSN=*.SMFDUMP.DUMPOUT`
where

- * means from this this job
- SMFDUMP is the name of the job step that created the dataset
- DUMPOUT is the dataset within the job step that holds the SMF records.

JCL to run the MQSMF program

When you have run the program check the output in //SYSOUT for messages like
error opening //LOG : EDC5037I The specified ddname was not found.
This indicates the //LOG statement is missing in the JCL.
The // statements in a bold font are new in this edition.

```
//S2 EXEC PGM=MQSMF,REGION=0M
//STEPLIB DD DISP=SHR,DSN=Your.load.library.name
//SMFIN    DD DISP=SHR,DSN=*.SMFDUMP.DUMPOUT
//SYSIN   DD *

Parameters – see below
//CHINIT  DD SYSOUT=*
//CHINCSV DD SYSOUT=*
//ADAPCSV DD SYSOUT=*
//SSLCSV  DD SYSOUT=*
//DISPCSV DD SYSOUT=*
//DNSCSV  DD SYSOUT=*
//ADAP   DD SYSOUT=*
//SSL     DD SYSOUT=*
//DISP    DD SYSOUT=*
//DNS     DD SYSOUT=*
//DCHS    DD SYSOUT=*
//DCHSCSV DD SYSOUT=*
//DCHSSUM DD SYSOUT=*
//CMESSAGE DD SYSOUT=*
//MESSAGE DD SYSOUT=*
//BUFF    DD SYSOUT=*
//BUFFCSV DD SYSOUT=*,DCB=(LRECL=200)
//DATA    DD SYSOUT=*
//QCPU    DD SYSOUT=*
//CF      DD SYSOUT=*
//CFCSV   DD SYSOUT=*
//DB2     DD SYSOUT=*
//EOJ     DD SYSOUT=*
//LOCK    DD SYSOUT=*
//LOG     DD SYSOUT=*
//LOGCSV  DD SYSOUT=*
//LOGBUSY DD SYSOUT=*
//MSGM    DD SYSOUT=*
//MSGMCSV DD SYSOUT=*
//QSUML   DD SYSOUT=*,DCB=(LRECL=200)
//QSUMS   DD SYSOUT=*,DCB=(LRECL=200)
//STGCSV  DD SYSOUT=*
//SMDS    DD SYSOUT=*
//TASKSUM DD SYSOUT=*
//TASK    DD SYSOUT=*
//TASKCSV DD SYSOUT=*
//TASKET  DD SYSOUT=*,DCB=(LRECL=200)
//TOPIC   DD SYSOUT=*
//STG     DD SYSOUT=*
//STGSUM  DD SYSOUT=*,DCB=(LRECL=200)
//
```

Where the DD statements are

- CHINIT page 18 Chinit statistics
- CHINCSV page 19 CHINCVS
- ADAP page 19 Adapter
- ADAPCSV page 20 Adapter CSV
- DISP page 21 Dispatcher tasks
- DISPCSV page 22 Dispatcher CSV
- DNSCSV page 23 Domain Name Server (DNS) CSV
- SSL page 24 SSL
- SSLCSV page 24 SSLCSV
- DNS page 23 Domain Name Server (DNS) task
- DCHS page 25 Channel accounting
- DCHSCSV page 27 Channel accounting CSV
- DCHSSUM page 28 Channel summary CSV
- CMESSAGE page 28 Chinit Messages(CMESSAGE)
- BUFF Page 49, Buffer pool statistics
- BUFFCSV Page 50, Buffer pool statistics in comma separated value format
- CF Page 31, Coupling facility statistics
- CFCSV Page 32, Coupling facility in comma separated value format
- DATA Page 51, Data manager statistics
- LOG Page 40, Log manager statistics
- LOGCSV Page 45, Log manager statistics in comma separated values
- LOGBUSY Page 46, Logbusy
- LOCK Page 54, Lock manager statistics
- MSGM Page 51, Message manager statistics (mq verbs)
- MSGMCSV Page 52, Message manager statistics (mq verbs) in comma separated value format
- STG Page 54, Storage manager statistics
- TOPIC Page 52, Topic manager statistics
- QSUML Page 39, Summary of queue usage - Local queues
- QSUMS Page 39, Summary of queue usage - Shared queues
- EOJ Page 51, Subsystem information
- DB2 Page 55, DB2 server information
- SMDS Page 33, Shared Message Data Set Statistics (SMDS)
- STGSUM Page 30, Address space level storage usage
- SYSOUT Output of warnings and problems identified in the data
- SYSPRINT Reports the parameters used, and task comments
- TASK Page 58, Detailed task data
- TASKCSV Page 73, Summarized task information
- TASLET Page 72, TaskElapsed Time
- TASKSUM Page 73, Task Summary

You will get data in QSUML, QSUMS, TASK,TASKCSV, and TASKSUM if TRACE(A)CLASS(3) is specified. You will get information on queues if the queue attribute ACCTQ is ON, either by setting it explicitly on a queue, or having ACCTQ(QMGR) and the QMGR attribute ACCTQ has the value ON.

The following section gives the parameters which can be passed too the program, through the //SYSIN ddname.

Record selection parameters

Value	Range	Default	meaning
CFStruct			CF Structure name
Channel			Only display channels beginning with the specified value.
Curdepth	0 to 999999999	0	If curdepth > 0 then information about the queues are only displayed when the the queue's *Curdepth maximum* is larger than the value specified. You can use this to display queue with unusually deep queues, such as transmission queues.
CICSTRAN			CICS transaction name. Selects transactions beginning with the specified value.
FirstRecord	0 to 999999999	0	specify starting record - 0 means not set
LastRecord	0 to 999999999	0	specify ending record - 0 means not set
Jobname			The prefix of jobs to be selected for example MQ02CHIN
QM			Select records by this queue manager
Queue			Select accounting records by this queue
QueueGetMB	0 to 9999	0	When this value is greater than 0, queues getting more than this value of MB of data, are reported in the QSUM* files
QueuePutMB	0 to 9999	0	When this value is greater than 0, queues putting more than this value of MB of data, are reported in the QSUM* files
StartTime			Use records if the start time is after hh:mm:ss
EndTime			Use records if the End time is before hh:mm:ss
zOS			Select record from this z/OS image

Interval for calculating data rates.

Value	Range	Default	meaning
SMF_Interval_time	1 to 999999999	60	The interval between SMF records in minutes. This is used when converting values to rates/second. For example the amount of data logged per second. This should be used when running MQ V7 without APAR PM61284, as this APAR adds the interval duration.

Thresholds for reporting out-of-line conditions in statistics

Value	Range	Default	meaning
BPIMW	1 to 999999999	1000	Report if buffer pool had more than this number of immediate writes
BPReadIOS	0 to 999999999	100	Report if buffer pool had more than this number of disk reads:Serious
BPReadIOW	0 to 999999999	0	Report if buffer pool had more than this number of disk reads:Warning
CFSTime	1 to 999999999	100	CF single response time in micro seconds
DeferredWriteTaskS	0 to 999999999	50	Number of DWT started-Serious
DeferredWriteTaskW	0 to 999999999	0	Number of DWT started-Warning
HighLogRateMB	1 to 9999	100	Specify high log rate in MB/Sec. A value larger than this will be reported as a high log rate. Many systems can achieve over 100MB/second. The value you get will depend on your hardware configuration.
OKLogRateMB	1 to 9999	50	OK log rate in MB/Sec
ReadLogBuffers	0 to 999999999	0	Report if the number of log records read exceeds this value

Thresholds for reporting out-of-line conditions in accounting data

Value	Range	Default	meaning
ApplicationLogMB	1 to 9999	1	Report in TASKCSV if application logged more than this value. Use this to select high use applications.
ApplicationCPUSec	1 to 9999	1	Report in TASKCSV if application used more than this CPU seconds. Use this to select high use applications.

BPGetPGetN	1 to 999999999	20	Report if ratio of get Old pages:get New pages exceeds this
CFSTime	1 to 999999999	100	CF single response time in micro seconds
CFSyncRT	1 to 999999999	10000	Report in MESSAGE if synchronous CF request greater than this value, in microseconds
CommitET	1 to 999999999	10000	If the commit time is longer than this value, produce message MQTASK12S, MQTASK13E or MQTASK14W
DB2Time	1 to 999999999	100	If the average time for DB2 request in micro seconds is greater than this value, produce message MQQ5ST11W, MQQ5ST12W, MQQ5ST13W or MQQ5ST14W
DNSLookupTime 1000	1 to 999999999	1000	Report if average time doing DNS lookup is longer than this value in microseconds.
ExitTime	1 to 999999999	10000	Report if the average ExitTime for a channel is longer than this value in microseconds.
HighLogRateMB	1 to 9999	100	Specify high log rate in MB/Sec. A value larger than this will be reported MQQJST10E in MESSAGE
Long_Open	50 to 999999999	1000	For IBM use only. Value to be considered a long open in micro seconds.
Long_Put	50 to 999999999	10000	If the average put times is greater than this value, produce message MQTASK08E or MQTASK09E .
LongLatchWait	50 to 999999999	1000	IBM use only. If the longest latch wait value is greater than this value report messages MQTASK15S or MQTASK15W
MaxDepth	0 to 999999999	0	If queue depth was greater than this value produce message MQTASK10W
NetTime	1 to 999999999	10000	Report if the average Nettime for a channel is longer than this value in microseconds.
OKLogRateMB	1 to 9999	50	OK log rate in MB/Sec
SSLDuration	1 to 999999999	10000	Report if the average duration for an SSL request a channel is longer than this value in microseconds.

Control what is displayed

Value	Range	Default	meaning
Debug			For IBM use only
Detail	0 to 20	5	Which level of detail 0 to 20. 20 provides the most detail
printHex			Debug value to print raw record in hex
IPLookup	0 to 1	0	displays the ip address for channels in format 0 – with ip address in numbers 1 – lookup the ip address to get the name for example MQPV.MQPH 9.20.4.5 or MQPV.MQPH winmvsca.ibm.com
MQ_LC_ALL		En_GB.IBM-285	Locale to be use in printing numbers. For example using MQ_LC_ALL 'En_GB.IBM-285' gives channel accounting data as 50,292,900 instead of the default 50292900. With De_AT.IBM-924 this value would be displayed as 50.250.324 See XL C/C++ XL C/C++ Programming Guide SC09-4765-12 for more information.

Processing Comma Separated Values

Some of the output files have data in Comma Separated Values(CSV) format.
You can have the output go to MVS data sets, or files in OMVS.
For example to create a file in USS.

```
//DNSCSV   DD PATH='/u/paice/DNS.csv',
//  PATHOPTS=(OWRONLY,OCREAT),PATHMODE=(SIRWXU,SIRGRP)
```

This file can then be imported into a spread sheet.

You can use cut and paste to copy the data into a spread sheet. For example with Open Office you can use paste, and use the options, *Separated by Comma, Space* and *Merge Delimiters.*

Output from the MQSMF program

Chinit statistics

In //CHINIT the channel initiator control information block contains basic information for the CHINIT, including:

- CHINIT job name (*qcctjobn*)
- QSG name if it is in a QSG (*qcctqsgn*)
- Number of current channels (*qcctnocc*)
- Number of active channels (*qcctnoac*)
- MAXCHL - maximum permitted current channels (*qcctmxcc*)
- ACTCHL - maximum permitted active channels (*qcctmxac*)
- TCPCHL - maximum permitted TCP/IP channels (*qcctmxtp*)
- LU62CHL - maximum permitted LU62 channels (*qcctmxlu*)
- Storage used by CHINIT (*qcctstus*)

You can use this information to see if the number of active channels is approaching the configured maximum value. Note that the number of current and active channels are the values when the record was created. So, between the two intervals there might have been more than this number of channels active.

Channel information from SMF data

Here is an example of channel information from SMF data:

```
MVCA,MQPV,2014/03/18,13:00:00,VRM:800,
From 2014/03/18,12:45:00.015222 to 2014/03/18,13:00:00.083630 duration
900.068408 seconds
Peak number used of current channels..........   4
Peak number used of active channels ..........   0
MAXCHL. Max allowed current channels..........9999
ACTCHL. Max allowed active channels...........9999
TCPCHL. Max allowed TCP/IP channels...........9999
LU62CHL. Max allowed LU62 channels.............. 200
Storage used by Chinit....................... 436MB
```

You can monitor the storage usage and see whether the value is trending upwards. If the total used is approaching the total storage available, you might be running out of storage, and so might not be able to support many more channels.

If the numbers of active current channels are tending towards the maximum number of channels, you might need to increase the maximum number of channels.

CHINCVS

This provides the same information as above but in a CSV format.

The column headings are

```
mvs,qm,qsg,date,time,QSG,HWMCCHL,HWMACHL,MaxCHL,MaxAct,MaxTCP,MaxLU,StgMB
```

Adapter

Adapter tasks

Example data

```
Task,Type,Requests,Busy %,     CPU used, CPU %,"avg CPU","avg ET"
   ,    ,        ,      ,      Seconds,      , uSeconds,uSeconds
   0,ADAP,  470297,  10.2,    41.290670,  4.6         88,     194
   1,ADAP,   13907,   0.6,     1.589428,  0.2        114,     365
   2,ADAP,    2517,   0.2,     0.185325,  0.0         74,     746
   3,ADAP,    1095,   0.1,     0.085774,  0.0         78,     907
   4,ADAP,     535,   0.1,     0.040743,  0.0         76,     947
   5,ADAP,     220,   0.0,     0.016228,  0.0         74,    1175
   6,ADAP,      82,   0.0,     0.005521,  0.0         67,    1786
   7,ADAP,      80,   0.0,     0.004248,  0.0         53,    1160
Summ,ADAP,  488733,   1.4,    43.217938,  0.6,        88,     205
```

The fields are calculated from:

- Duration: qwhs.qwhsdurn
- Requests: qctreqn
- Busy %: qcteltm and duration
- CPU used: qctcptm
- CPU %: qctcptm and duration
- Average CPU: qctcptm and qctreqn average
- ET: qcteltm and qctreqn

This example shows that there were eight adapter tasks.

Adapter number 0

- Processed the majority of the requests (470297 out of 488733)
- Was busy 10.2% of the interval
- Used 41.3 seconds of CPU

Overall

The average CPU per request was 88 microseconds of CPU and took 205 microseconds

The times are in STCK values. To convert these to microseconds divide by 4096 (which removes the bottom 12 bits).

The adapters process MQ requests. Some of these requests might wait, for example, for log I/O

during a commit, so the average Elapsed Time per request has little meaning.

When a WebSphere MQ request is made the first free adapter task is used.

- If there is at least one adapter that has been little used (less than 1% busy), you have enough adapters.
- If at least one adapter was not used, you have enough adapters defined.
- If all the adapters were used, you might need to allocate more adapters.
- If all of the adapters were used, and they were all busy for most of the interval, you need to allocate more adapters.

You can use the ALTER QMGR CHIADAPS() command to change the number of adapters used. Any changes come into effect the next time the CHINIT is started.
Attention: If there are too many adapters acting on a small set of queues, you might get contention within the queue manager.

Understanding the data

Usually the first adapter processes most of the requests, and the other adapters process fewer and fewer requests. The first few adapters may be busy for most of the period (close to 100 %busy).

In a system constrained for CPU the data looks different for example

```
Task,Type,Requests,Busy %,     CPU used, CPU %,"avg CPU","avg ET"
    ,    ,        ,      ,      Seconds,      , uSeconds,uSeconds
   0,ADAP,   30005,  67.1,     2.802067,  6.0,        93,    1051
   1,ADAP,   27121,  64.1,     2.561604,  5.5,        94,    1111
   2,ADAP,   25668,  61.7,     2.418879,  5.1,        94,    1131
   3,ADAP,   24154,  58.5,     2.269495,  4.8,        94,    1138
   4,ADAP,   22698,  55.2,     2.127903,  4.5,        94,    1142
   5,ADAP,   20906,  51.4,     1.956664,  4.2,        94,    1156
   6,ADAP,   19094,  47.1,     1.781381,  3.8,        93,    1160
   7,ADAP,   16827,  42.3,     1.568889,  3.3,        93,    1181
Summ,ADAP,  186473,  55.9,    17.486883,  4.7,        94,     100
```

Here, none of the task are close to 100% busy.
The adapter task records how busy it is by calculating the start and end times of the request. If there is a delay in z/OS dispatching the TCB then the adapter task will appear busy. So it looks like the first adapter was not dispatched 33 % of the time (100-67.1%).
If you see this pattern of data then check the chinit address space is not constrained for CPU. Also check the dispatcher data. See below.

Adapter CSV

This is the same information as the adapter section, but in CSV format, which can imported into a spread sheet

Dispatcher tasks

Example data

```
Task,Type,Requests,Busy %,     CPU used, CPU %,"avg CPU","avg ET"
    ,    ,        ,      ,      Seconds,      , uSeconds,uSeconds
   0,DISP,   26587,   0.4,     0.592463,  0.1         22,     127
   1,DISP,   26963,   0.3,     0.588092,  0.1         22,     112
   2,DISP,  864329,   2.7,     2.545668,  0.3          3,      28
   3,DISP,   26875,   0.4,     0.590825,  0.1         22,     120
   4,DISP,   26874,   0.4,     0.603285,  0.1         22,     123

Summ,DISP,  971628,   0.8,     4.920332,  0.1,         5,      38
```

The example data shows that there were five dispatchers. A channel is associated with a dispatcher, and channels are distributed across all the dispatchers. This example shows that one dispatcher is processing more requests than other dispatchers. This is normal, as some channels might stop, so the dispatcher is processing fewer channels, and some channels can be busier than others.

- 4.9 seconds of CPU were used by the dispatchers.
- The average request used 5 microseconds of CPU and took 38 microseconds elapsed time.
- The average CPU used per request depends on the message traffic, for example, bigger messages use more CPU than smaller messages.

The fields are calculated from:

- Duration: qwhs.qwhsdurn
- Requests : qctreqn
- Busy %: qcteltm and duration
- CPU used: qctcptm
- CPU %: qctcptm and duration
- Average CPU: qctcptm and qctreqn
- Average ET: qcteltm and qctreqn

Usually, the number of dispatchers should be less than, or equal to, the number of processors in the LPAR. If you have more dispatchers than processors in the LPAR they might compete for CPU resources.

Channels have an affinity to a dispatcher, so you might find that some dispatchers process many more requests than another dispatcher.

You can use the ALTER QMGR CHIDISPS() command to change the number of dispatchers used. Any change comes into effect the next time the CHINIT is started.

Understanding the data.

A dispatcher is used to send and receive data over a communications network, and this is not usually dependent on external events. The average elapsed time should, therefore, be close to the average CPU time used. If the CHINIT is delayed due to lack of CPU, then the ratio of average Elapsed Time to average CPU time is much larger, compared to when the CHINIT is not delayed

for CPU.
In a test systems which was constrained for CPU the elapsed time was more than 10 times the CPU used.

Dispatcher CSV

This is the same information as the dispatcher section, but in CSV format, which can imported into a spread sheet

Domain Name Server (DNS) task

The data is similar to the adapter and dispatcher tasks, but there is additional information on the longest request at the end of the record.

On TCB for the DNS requests there is

```
longest ,date      ,time
uSeconds,          ,
     463,2014/03/18,12:56:33.987671
```

The longest DNS request took 463 microseconds elapsed time, and this occurred at 12:56:33 local time.

The fields are calculated from:

- Duration: qwhs.qwhsdurn
- Requests : qctreqn
- Busy %: qcteltm and duration
- CPU used: qctcptm
- CPU %: qctcptm and duration
- Average CPU: qctcptm and qctreqn
- Average ET: qcteltm and qctreqn
- Longest: qctlgdu
- Longest at: qctlgtm

The DNS is used when a channel starts, to convert an ouput connection name to an IP address, and from an input IP address to a connection name.

The DNS task can go out of your enterprise to look up a value. If the average elapsed time is significantly more than the average CPU time used, you might have some long requests.

If the value of the longest request time is unacceptable you should work with your network team to investigate why you are having long requests. It might be that you have an invalid name in your conname.

If the DNS task is busy for 25% of the duration, consider investigating the cause further.

Note: There are requests to the DNS task that are not DNS lookups, so you might have the number of requests being greater than zero – but no longest request information.

Domain Name Server (DNS) CSV

This is the same information as the DNS section above, but in CSV format, which can imported into a spread sheet.

SSL

This is the same information as the Dispatcher section above. It has additional information for each task.

```
Task,... longest ,date      ,time
    ,... uSeconds,          ,
   0,... 8864,2014/03/18,12:46:40.237697
   1,... 4714,2014/03/18,12:46:18.938022
   2,... 7273,2014/03/18,12:46:35.358145
   3,... 13164,2014/03/18,12:46:44.514045
   4,...22438,2014/03/18,12:46:22.134123
Summ,...22438,2014/03/18,12:46:22.134123
```

The longest request was for SSL task 4, took 22,438 microseconds, and occurred at 12:46:22.134123 local time. The overall longest value is displayed in the Summ record.

The fields are calculated from:

- Duration: qwhs.qwhsdurn
- Requests : qctreqn
- Busy %: qcteltm and duration
- CPU used: qctcptm
- CPU %: qctcptm and duration
- Average CPU: qctcptm and qctreqn
- Average ET: qcteltm and qctreqn
- Longest: qctlsdu longest at: qctlstm

A running channel is associated with an SSL task, in a similar way that a channel is associated with a dispatcher. The SSL tasks can use the cryptographic coprocessors available to the LPAR. So, the elapsed time can include time spent on a coprocessor. You should monitor the average elapsed time throughout the day. If this time increases significantly during peak periods you should work with your MVS systems programmers, as your coprocessors might be over utilized.

If the SSL tasks are busy for a significant proportion of the interval, increasing the number of SSL tasks might help. If the SSL tasks are waiting for external resources such as a coprocesor, increasing the number of SSL tasks may have little effect.

You can use the ALTER QMGR SSLTASKS() command to change the number of SSL tasks used. Any changes come into effect the next time the CHINIT is started.

SSLCSV

This record has the information from the Summ section of the SSL records above

Channel accounting

This information is based on the output from Display Channel Status.

```
Connection name               winmvsca.hursley.ibm.com
Connection name               9.20.4.159
Channel disp                  PRIVATE
Channel type                  SENDER
Channel status                RUNNING
Channel STATCHL               HIGH
Remote qmgr/app               MQPV
Channel started date & time  2014/06/04,08:05:50
Channel status collect time  2014/06/04,08:08:16
Last msg time                2014/06/04,08:07:50
Active for                       145 seconds
Batch size                            50
Messages/batch                      48.8
Number of messages                33,896
Number of persistent messages     33,896
Number of batches                    695
Number of full batches               601
Number of partial batches             94
Buffers sent                      33,992
Buffers received                     697
Xmitq empty count                      0
Message data                  120,263,008   114 MB
Persistent message data       120,263,008   114 MB
Non persistent message data             0     0  B
Total bytes sent              120,266,176   114 MB
Total bytes received               19,996    19 KB
Bytes received/Batch                   28    28  B
Bytes sent/Batch                  173,044   168 KB
Batches/Second                         4
Bytes received/message                 0     0  B
Bytes sent/message                 3,548     3 KB
Bytes received/second                137   137  B/sec
Bytes sent/second                829,421   809 KB/sec
Compression rate                       0
Exit time average                      0 uSec
Net time average                     116 uSec
Net time min                          71 uSec
Net time max                      14,481 uSec
Net time max date&time           2014/06/04,08:07:41
```

Field name	**Example value**		`Field name`
Connection name	winmvsca.hursley ibm.com		`C run time function inet_pton(qcstcnnm)`
Connection name	9.20.4.159		qcstcnnm
Channel disp	PRIVATE		qcstchdp values MQCHLD_*
Channel type	SENDER		qcstchty values MQCHT_*

Channel status	RUNNING		qcstchst values MQCHS_
Channel STATCHL	HIGH		qcststcl values MQMON_*
Remote qmgr/app	MQPV		qcstrqmn
Channel started date & time	014/06/04,08:05:50		qcststrt
Channel stopped time			qcstludt
Channel status collect time	014/06/04,08:08:16		qcstcltm
Last msg time	014/06/04,08:07:50		qcstlmst
Active for	145 seconds		Start time = later of channel start time qcststrt, and interval start time End time = earlier of channel ended time qcstludt, and end of interval qwhsstck
Batch size	50		qcstcbsz
Messages/batch	48.8		qcstnmsg/qcstbatc
Number of MQI requests (for svrconn)			qcstnmsg
Number of messages	33896		qcstnmsg
Number of persistent messages	33896		qcstnpmg
Number of batches	695		qcstbatc
Number of full batches	601		qcstfuba
Number of partial batches	94		qcstbatc-qcstfuba
Buffers sent	33,992		qcstbfst
Buffers received	697		qcstbfrc
Xmitq empty count	0		qcstqetc
Current shared connections			qcstcscv
Message data	120263008	114 MB	qcstnbyt
Persistent message data	120263008	114 MB	qcstnpby
Non persistent message data	0	0 B	qcstnbyt - qcstnpby
Total bytes sent	120266176	114 MB	qcstbyst
Total bytes received	19996	19 KB	qcstbyrc
Bytes received/Batch	28	28 B	qcstbyrc/ qcstbatc
Bytes sent/Batch	173044	168 KB	qcstbyst/ qcstbatc

Batches/Second	4		qcstbatc/active time
Bytes received/message	0	0 B	qcstbyrc/ qcstnmsg
Bytes sent/message	3548	3 KB	qcstbyst/ qcstnmsg
Bytes received/second	137	137 B/sec	qcstbyrc/active time
Bytes sent/second	829421	809 KB/sec	qcstbyst/active time
Compression rate	0		qcstcpra
Exit time average	0 uSec		qcstetav
Exit time min			qcstetmn
Exit time max			qcstetmx
Exit time max date&time			qcstetmx
Net time average	116 uSec		qcstntav
Net time min	71 uSec		qcstntmn
Net time max	14,481 uSec		qcstntmx
Net time max date&time	2014/06/04,08:07:41		qcstntdt
Put retry count			qcstptrc See below

Note: qcstbyst and qcstbyrc are the values from the Display Channel Status command. The values in these fields wrap at 999,999,999. You can get the true number of bytes sent by adding 999,999,999 to this value until the value exceeds the qcstnbyt value

The Put retry count value is the number of times a receiver channel tried to put a message to a queue and had problems.
For example
+CSQX038E MQPV CSQXRESP Unable to put message to MPQAZ, MQCC=2
MQRC=2192 (MQRC_PAGESET_FULL)
+CSQX565E MQPV CSQXRESP No dead-letter queue for MQPV, channel MQPH.MQPV

The put request is retried. See channel attributes MRTMR and MRRTY for how many times and how often it retries. During this time the channel is paused and is not processing any messages.

Channel accounting CSV

This is the same information as the Channel section, but in CSV format, which can imported into a spread sheet. The columns are

MVS	z/OS system name
MQ	Queue Manager name
date	Date the record was created
time	time the record was created
channelType	Sender etc

ChannelName
BSZ Negotiated batch size
ABSZ Achieved batch size
Bytes/sec

Channel summary CSV

This provides a summary of the activity for different channel types.

For example

```
MVS,MQ,date,time,VRM,channelType,count,Persistent,NonPersistent,'P/Sec','NP/Sec'
MVCA,MQPV,2014/06/30,11:30:00,VRM:800,RECEIVER,2,75720,0,3786,0
MVCA,MQPV,2014/06/30,11:30:00,VRM:800,total,2,75720,0,3786,0
MVCA,MQPH,2014/06/30,11:30:00,VRM:800,SENDER,2,75720,0,2611,0
MVCA,MQPH,2014/06/30,11:30:00,VRM:800,total,2,75720,0,2611,0
MVCA,MQPH,2014/06/30,11:34:04,VRM:800,SENDER,23,86237508,0,559983,0
MVCA,MQPH,2014/06/30,11:34:04,VRM:800,total,23,86237508,0,559983,0
```

Chinit Messages(CMESSAGE)

This file contains information about the Chint SMF Data.

Checks are made with user specified values, such as SSL Duration. See the value that can be specified in Record selection parameters on page 13.

The number in parenthesis () before the message is when the message is displayed. The message is displayed if the value of Detail in the input parameters is >= the value in ().

The messages text is a paraphrase of the message, to make it easier to read.

(0) MQCHIN000S The high water mark of the number of active channels >90 % of max channels.

(10) MQCHIN001W The high water mark of the number of active channels >50 % of max channels.

(10) MQCHIN001I The high water mark as a percentage of the max channels.

(10) MQCHIN006S Longest SSL request >> specified SSL duration (SSLDuration).

(0)MQCHIN011S High water mark of active channels nn% > 90% of max active channels.

(10)MQCHIN003W High water mark of active channels nn% > 50% of max active channels.

(10)MQCHIN013I High water mark of active channels nn% max active channels.

(0) MQCHIN007I Dispatcher task is nn% busy on average .

(0) MQCHIN008I Adapter task is nn% busy on average.

(0) MQCHIN009I SSL task is nn% busy on average.

(0) MQCHIN006S Longest SSL request (n uSeconds) >> specified SSL specified duration(nn uSeconds).

The longest SSL request was greater than 10 time the value of SSLDuration .

(0) MQCHIN005W Longest SSL request (n uSeconds) > specified SSL specified duration(nn uSeconds).

The longest SSL request was greater than 10 value of SSLDuration .

(0) MQCHIN007I DNS task is nn% busy on average.

(0) MQCHIN004S Longest DNS request (n uSeconds) >> specified DNS duration(nn uSeconds).

The longest DNS request was greater than 10 time the value of DNSLookupTime.

Investigate why the DNS request took so long. An invalid address may have been specified, and the request may have been routed out of your domain.

(0) MQCHIN005W Longest DNS request (n uSeconds) specified SSL specified duration(nn uSeconds).

Address space level storage usage

The address space storage usage statistics give information about virtual storage and real storage used by the queue manager.

The output is displayed in the ddname //STGSUM.
For the details about the fields see the layout for CSQDQSRS in SCSQC370(CSQDSMFC).

```
>16MB Used   397 MB Free  1080 MB %used  26 delta    4 MB
Bar   Used   237 MB           %used   0
Real  Used                  1010 MB
 why limited:Set in the JCL
```

`>16MB` shows 397 MB of the region's virtual storage has been used. There is 1080MB left in the region. The amount used is 26% of the total virtual storage. Since the last storage SMF record there has been a 4MB increase in virtual storage usage.

`Bar` shows the storage usage above the bar.

`Real` shows the real storage usage.

`why limited`: This what limits the 31 bit storage. The value (raxlvmemlims) is extracted from a z/OS control block. The common values of this are:

- Set by SMF or SMF default
- Set in the JCL
- Unlimited Region=0
- Set by IEFUSI
- Set by Auth interface

Other information displayed is likely to be of use to IBM only.

Additional messages produced in MESSAGE output file.

MQVSTG01E virtual storage usage > 95%

When: 31 bit storage use is > 95% of the available storage use.
Action: Review storage usage. Consider making buffer pools smaller.
Using detail(20) will provide additional information.

MQVSTG02W virtual storage usage > 90%

When: 31 bit storage use is > 90% of the available storage use.
Action: Review storage usage. If start to get this message, you should monitor the storage usage. If your storage continues to increase, you should start reviewing how you are using your buffer pools and perhaps reduce the size of them.

Coupling Facility statistics

The CF statistics give information about the Coupling Facility usage.

When a message is put or got from a queue the request to the CF has a single update. During a commit, the request to the CF may change several messages, so this is counted as a multiple request.

The data is displayed in the ddname //CF.
See the record layout in SCSQMACS(CSQDQEST) for interpretation of these fields.

```
MVCA MQ7B 2013/01/08 11:25:44 VRM:710
  APP1        , Structure-fulls      0
    Single     1000, avg et in uS    14, Retries      0
    Multiple     24, avg et in uS  1185, Retries      0
    Max entries      1033,  Max elements      2048
```

Where

`APP1`	The name of the structure
`Structure-fulls`	The number of times the structure filled
`Single`	The number of requests where there was a single request in the CF request
`avg et in Us`	The average elapsed time where there was a single request in the CF request, in microseconds
`Retries`	The number of times a request was retried
`Multiple`	The number of requests where there were multiple request in the CF request
`avg et in uS`	The average elapsed time where there were multiple request in the CF request, in microseconds
`Retries`	The number of times a request was retried, where there were multiple request in the CF request
`Max entries`	The maximum number of entries used in the structure
`Max elements`	The maximum number of elements used in the structure.

Additional messages produced in MESSAGE output file.

Message: MQQEST00E QEST ... structure sss full n times

When: qestsful>0
Reason: The CF Structure has reached its capacity.
Action: Investigate to see if this is a short term problem, or a longer term problem. If this is a long term problem, you will need to increase the size of the CF structure.

Message: MQQEST01S QEST ... structure sss extremely long average CF response time n uS

When: The average CF Single response time is > 100 * value specified in CFStime parameter.
Reason: The average response time of the single requests is taking a long time.
Action: Review the performance of the Coupling Facility. For example there may be a remote

Coupling Facility. You may have specified a value of CFStime which is unrealistic.

Message: MQQEST02E QEST ... structure sss very long average response time n uS

When: The average CF Single response time is > 10 * value specified in CFStime parameter.
Reason: The average response time of the single requests is taking a long time.
Action: Review the performance of the Coupling Facility. For example there may be a remote Coupling Facility. Determine if this occurred a time of peak workload.
You may have specified a value of CFStime which is unrealistic.

Message: MQQEST03W QEST ... structure sss long average response time n uS

When: The average CF Single response time is greater than the value specified in CFStime parameter.
Reason: The average response time of the single requests is taking a long time.
Action: Review the performance of the Coupling Facility. For example there may be a remote Coupling Facility. Determine if this occurred a time of peak workload.
You may have specified a value of CFStime which is unrealistic.
It is acceptable to get message MQQEST03W, but you can use it as a warning if the structure response time increases.

Coupling Facility CSV

The CF statistics give information about the Coupling Facility usage.
The data reported in the ddname //CFCSV is Coupling Facility in a single line, in comma separated values.

```
'MVS','QM','DATE','TIME','Structure','Full','Max entries','Max elements','avg S','avg M','Num S','Num M'
MVCA,MQ7A,2013/02/09,14:18:39,CSQ_ADMIN   ,     0,   345,   390,925,1287,1999,5
MVCA,MQ7A,2013/02/09,14:18:39,APP1        ,     0,    35,    68,1027,1112,3000,999
```

Where the fields are

MVS	The MVS system ID
QM	The queue manager name
DATE	The date in YYYY/MM/DD format
TIME	The time in hh:mm:ss
Structure	The name of the structure
Full	The number of times the structure was full
Max entries	The maximum number of entries in time period
Max elements	The maximum number of elements in the time period
avg S	The average CF response time for single requests
avg M	The average CF response time when there are multiple data requests in a single CF request
Num S	The number of single requests
Num M	The number of requests when there are multiple data requests in a single CF request.

Shared Message Data Set Statistics (SMDS)

Shared Queue messages can be offloaded from the coupling facility to Shared Message Data Set(SMDS).

The output is displayed in the ddname //SMDS. For the details about the fields see SCSQMACS(CSQDQESD).

SMDS has a lower CPU cost, and higher throughput than storing the messages in DB2.
When SMDS is used, each queue manager in the QSG has its own dataset for storing messages, and has read access to the data sets from the other queue managers in the QSG. There is zero or one SMDS data set per queue manager for a CF Structure.

When a message is put, and it goes to SMDS, then a buffer is used to write to the SMDS. The buffer is freed before the put request returns to the application. So the buffer is used only for the duration of the I/O request. The number of buffers available for use can be configured . The default buffer size is 256K. Once set, the buffer size cannot be changed. When messages are put which are larger than this, multiple buffers are used. An application may have to wait for a buffer, and will have to wait if the request requires I/O. During a get the data may already be in a buffer. In this case there is no I/O to the SMDS.

The highlighted lines in the report below are used as headings in the following sections.

```
MVCA,MQ7A,2013/02/23,15:49:21,VRM:710,
 CF manager shared message data set (SMDS) statistics
  Structure :   2,  Name APP1
    SMDS space management statistics:
    SMDS space management usage:
      Messages in data set          27827        highest      27827
      Total blocks                  22914
        Space map blocks                1
        Message data blocks         22913
        Data blocks used            11306 ( 49%) highest      11306 ( 49%)
        Data blocks free            11607 ( 51%) lowest       11607 ( 51%)
    SMDS space management activity:
      Action        Messages     4K pages
      Allocated        27827       723502
      No space             0
      Released             0            0
      Reallocated          0            0
      Cleaned up           0            0
SMDS buffer pool statistics:
  SMDS buffer pool usage:
    Buffer size (DSBLOCK)           256K
    Total buffers                     1
    Buffers in use                    1 (100%)  highest      1 (100%)
      Shared buffers                  1
    Buffers free                      0 (  0%)  lowest       0 (  0%)
      Saved buffers                   0
      Empty buffers                   0
    Waiting request queues
      For free buffer                 1         highest      1
      For busy buffer                 0         highest      0
  SMDS buffer pool activity:
    Acquired buffers              39131
```

```
      Got valid buffer                       0 (  0%)
      Got matching, empty buffer             0 (  0%)
      Got free, empty buffer                 1 (  0%)
      Stole a saved buffer                39130 ( 99%)
    No buffer available           11305
    Waited for free buffer         6217 ( 16%) avg time 0.004568s
    Waited for busy buffer            0 (  0%) avg time 0.000000s
    Buffer read issued             1406
      Data already valid              0 (  0%)
      Data partly valid               0 (  0%)
      Data read from disk          1406 (100%)
    Freed valid buffer            39130
    Marked buffer deleted             0
    Buffer write issued           39130
SMDS I/O statistics:
  SMDS data set usage:
    High allocated CI           1466496
    High formatted CI           1466496
    Control interval size          4096
    Control area size            589824
  SMDS I/O activity:
    Type           Requests    4K pages pages/req  avg I/O time  avg wait time
    Format                0           0    0.0        0.000000s      0.000000s
    Write             39130      723467   18.5        0.000676s      0.000656s
    Read (local)       1406       26000   18.5        0.000438s      0.000336s
    Read (Other)      39107      723034   18.5        0.000663s      0.000646s
```

Where the records in the output file are described below. The highlighted lines in the report are headings below.

Structure : 2, Name APP1

This identifies the structure.
Field name used QESDSTRN,QESDSTR.

SMDS space management statistics:

The space management statistics give you information on the usage and activity of the SMDS data set owned by this queue manager.

SMDS space management usage:

This section gives information on the number of messages in the the dataset, how many blocks are in use, and how many blocks are available.

```
    SMDS space management usage:
      Messages in data set          27827        highest      27827
```

This is number of messages in the data set, when the SMF record was created, and the highest in the interval.
Field names used QESDSMMC, QESDSMMM.

```
      Total blocks                  22914
        Space map blocks                1
        Message data blocks         22913
        Data blocks used            11306 ( 49%) highest      11306 ( 49%)
        Data blocks free            11607 ( 51%) lowest       11607 ( 51%)
```

This is information about the number of records in the amount of space used. When 100 1MB messages were put to the queue 402 blocks were used. Each block/buffer size is 256KB

(DSBLOCK attribute of CFSTRUCT) see below. 4 Blocks per 1MB message * 100 messages = 400 blocks.
Field names used QESDSMBT;QESDSMBS;QESDSMBD; QESDSMMC, QESDSMMM; QESDSMBU, (QESDSMBU,QESDSMBD),QESDSMMU,QESDSMMU,QESDSMBD;QESDSMBF (QESDSMBF, QESDSMBD), QESDSMMF,(QESDSMMF,QESDSMBD).

SMDS space management activity

This section gives information on the activity of the SMDS. The usage section above shows how many blocks have been used. This section tells you how many times the blocks were used.

```
SMDS space management activity:
  Action        Messages     4K pages
  Allocated        27827       723502
```

This is the number of requests to use a message or a page. A buffer and page can be reused may times.
Field names used: QESDSMAR,QESDSMAP.

```
  No space             0
```

This number of times the SMDS had no space. This should be 0 in normal usage.
Field name used QESDSMFL.

```
  Released             0            0
  Reallocated          0            0
  Cleaned up           0            0
```

Field names used QESDSMFR, QESDSMFP; QESDSMRR, QESDSMRP;QESDSMCR, QESDSMCP.

SMDS buffer pool statistics:

The queue manager has a number of buffers to access the SMDS. This section gives information on the number of buffers and how often they were used.

SMDS buffer pool usage:

This section gives the size of the buffers (the block size) and how many buffers have been used.

```
SMDS buffer pool usage:
  Buffer size (DSBLOCK)          256K
```

This is the size of the buffers, and so the block size of the data.
Field name used: QESDBFSZ/1024.

```
  Total buffers                    1
```

This is the number of buffers allocated. One buffer was specified to produce some of the wait conditions below. You would normally have enough buffers, so that you application did not have to wait for a buffer. Using a large number of buffers can uses a lot of of MVS auxiliary storage and MVS real storage.
Field name used QESDBFTO.

```
  Buffers in use                   1 (100%)  highest      1 (100%)
    Shared buffers                 1
  Buffers free                     0 (  0%)  lowest       0 (  0%)
    Saved buffers                  0
    Empty buffers                  0
  Waiting request queues
    For free buffer                1         highest      1
```

```
      For busy buffer                  0         highest      0
```
Field name used: QESDBFFS+QESDBFFE, (QESDBFFS+QESDBFFE)/QESDBFTO, QESDBFMF, 100 * QESDBFMF/QESDBFTO; QESDBFUS; QESDBFFS+QESDBFFE, (QESDBFFS+QESDBFFE)/QESDBFTO, QESDBFMF, QESDBFMF/QESDBFTO; QESDBFFS; QESDBFFE; QESDBFPW,QESDBFMP; QESDBFBW,QESDBFMB.

SMDS buffer pool activity:

This section gives information on the activity of the buffers. The buffer pool usage section above shows how many buffers were used. This section displays how many times the buffers were used.

```
  SMDS buffer pool activity:
    Acquired buffers              39131
      Got valid buffer                        0 (  0%)
```
This is the number of buffers when a get was issued, and the data was in the buffer, and so there was no need to get the data from the SMDS.
Field name used: QESDBFGB; QESDBFGV, QESDBFGV/QESDBFGB.

```
      Got matching, empty buffer              0 (  0%)
      Got free, empty buffer                  1 (  0%)
```
This number of times a buffer was obtained which was free, with no data in it. This occurs when the queue manager is started, and so the buffers are empty.
Field name used: QESDBFGM, QESDBFGM/QESDBFGB;QESDBFGF,QESDBFGF/QESDBFGB.
QESDBFGL, QESDBFGL/QESDBFGB.

```
      Stole a saved buffer               39130 ( 99%)
```
This number of times there were no free buffers, and so an existing buffer was used.
Field names used: QESDBFGL, QESDBFGL/QESDBFGB.

```
    No buffer available           11305
```
This number of times there were no buffers, available because they were all in use. You should increase the number of buffers.
Field name used: QESDBFGN.

```
    Waited for free buffer         6217 ( 16%) avg time 0.004568s
```
This number of times there were no free buffers, and the average wait time until a buffer was available.
Field names used: QESDBFWP,(QESDBFWP,QESDBFGB),QESDBFPT/QESDBFWP.

```
    Waited for busy buffer             0 (  0%) avg time 0.000000s
```
This number of times there was a request for an existing buffer's content, but the buffer was in use, for example while it was being written to the SMDS, and the average wait time for this buffer.
Field names used: QESDBFWB,(QESDBFWB,QESDBFGB),QESDBFBT/QESDBFBW.

```
    Buffer read issued             1406
      Data already valid              0 (  0%)
      Data partly valid               0 (  0%)
      Data read from disk          1406 (100%)
```
This number of times there was a request to get a message from the queue manager that owns the SMDS.
The data shows all records were read from the SMDS, and none were already available in buffers
Field name used: QESDBFRR;QESDBFRS,QESDBFRS/QESDBFRR;QESDBFRP, QESDBFRP/QESDBFRR;QESDBFRR-QESDBFRS-QESDBFRP, (QESDBFRR - QESDBFRS - QESDBFRP)/QESDBFRR;

```
    Freed valid buffer            39130
```
This is the number of times buffer with valid content was reused.
Field names used: QESDBFFB.

```
    Marked buffer deleted             0
    Buffer write issued           39130
```
This is the number of times a buffer was written to the SMDS.
Field names used: QESDBFDB;QESDBFWR.

SMDS I/O statistics:

The section reports on the I/O statistics to the SMDS owner by this queue manager, and the read activity to the SMDS from other queue managers.

SMDS data set usage:

The section report on the highest use record in the dataset, the size of the data pages. The data page (Control Interval), size is always 4096 bytes. The Control area size is the DSBlock size on the Alter or Define CFSTRUCT command.
This is for the R/W SMDS owned by this queue manager.

```
SMDS I/O statistics:
  SMDS data set usage:
    High allocated CI          1466496
    High formatted CI          1466496
    Control interval size         4096
    Control area size           589824
```
Field names used: QESDIOHA, QESDIOHU, QESDIOCI, QESDIOCA;

SMDS I/O activity:

This section reports on the I/O activity when formatting a new extent, for writing records, and for reading records, for the R/W SMDS for this queue manager.

```
  SMDS I/O activity:
    Type           Requests    4K pages pages/req  avg I/O time  avg wait time
    Format                0           0    0.0        0.000000s      0.000000s
    Write             39130      723467   18.5        0.000676s      0.000656s
    Read (local)       1406       26000   18.5        0.000438s      0.000336s
```
This is information about the R/W SMDS for the queue manager. The records are

- Format- if the SMDS expanded in size, then it extends the data set, and formats it
- Write - data is written to the data set
- Read - data is read from the data set

The columns are

- Requests – the number of I/O requests
- 4K pages – the number of 4K pages written
- pages/req – the number of pages per request
- avg I/O time – the average I/O time for the request.
- avg wait time – the average time the request had to wait before the I/O could be started. If this value is larger than 0, this indicates there were not enough buffers.

Field names used:
QESDIOFR, QESDIOFP, QESDIOFR/QESDIOFP,QESDIOFT/QESDIOFR, QESDIOFW,QESDIOFR;

QESDIOWR, QESDIOWP, QESDIOWR/QESDIOWP,QESDIOWT/QESDIOWR, QESDIOWW,QESDIOWR;
QESDIORR, QESDIORP, QESDIORR/QESDIORP,QESDIORT/QESDIORR, QESDIORW,QESDIORR;

When reading from an SMDS belonging to another queue mangers the following information is provided. Read (other). The data has the same interpretation as for the local SMDS data set. The data is accumulated for all of the other SMDS for the other systems, it is not available for individual data sets.

```
Type           Requests    4K pages pages/req  avg I/O time  avg wait time
Read (Other)      39107      723034   18.5        0.000663s      0.000646s
```

Field names used:
QESDIOOR, QESDIOOP, QESDIOOR/QESDIOOP,QESDIOOT/QESDIOOR, QESDIOFW,QESDIOOR;

QSUML – Queue summary information for local queues

The QSUML data is a summary of the queue usage over time, for local queues

```
Date      ,Time    ,Qmgr,Queue,   Count, PS,BP,Put MB ,Get MB ,!,ValidPut,ValidGet,getpsn,MaxQDepth,
2013/01/14,15:00:00,MQPA,CP0000   ,2134, 2,  2,1.8e+06,1.8e+06,!,1.8e+03, 1.8e+03,     0,       50
2013/01/14,15:00:00,MQPA,CP0001   ,  17, 1,  1,1.4e+04,1.4e+04,!,     14,       14,     0,        2
2013/01/14,15:00:00,MQPA,CP0002   ,  17, 1,  1,1.4e+04,1.4e+04,!,     14,       14,     0,        2
2013/01/14,16:00:00,MQPA,CP0000   ,2134, 2,  2,3.3e+05,3.3e+05,!,3.3e+02, 3.3e+02,     0,       50
2013/01/14,16:00:00,MQPA,CP0001   ,  17, 1,  1,2.7e+03,2.7e+03,!,      3,        3,     0,        2
```

Where the fields are

Date	in format YYYY/MM/DD
Time	is either the queue open time (if the open was in the SMF interval) or the interval start time from the WTAS task record. The time is then rounded down to the nearest hour.
QMGR	is the queue manager name
Queue	is the queue name
Count	is the number of queue records processed.
PS	is the page set number
BP	is the buffer pool
Put MB	is the amount of data put to the queue in MB
Get MB	is the amount of data got from the queue inMB
ValidPut	is the number of valid put requests
ValidGet	is the number of valid get requests
Getpsn	is the number of gets from the page set
MaxQDepth	is the maximum depth of the queue.

QSUMS – Queue summary information for Shared queues

The QSUMS data is a summary of the queue usage over time, for shared queues

```
Date      ,Time    ,Qmgr,Queue,Count,Structure   ,Put MB,Get MB,!,ValidPut,ValidGet,MaxQDepth,
2013/01/15,13:00:00,MQPA,SQ1  ,    3,APP1        , 3e+03, 2e+03,!        3,     2,         1,
```

Where the fields are

Date	in format YYYY/MM/DD
Time	is either the queue open time (if the open was in the SMF interval) or the interval start time from the WTAS task record. The time is then rounded down to the nearest hour.
Qmgr	is the queue manager name
Queue	is the queue name
Count	is the number of queue records processed.
Structure	is the CF Structure name
Put MB	is the amount of data put to the queue in MB
Get MB	is the amount of data got from the queue in MB
ValidPut	is the number of valid put requests
ValidGet	is the number of valid get requests
MaxQDepth	is the maximum depth of the queue.

Log statistics

The log statistics give information about the amount of data written to the log datasets.

You can use this section

1. To determine if you are reaching the limit at which the queue manager can write data to the log data sets.
2. For regular checks of the logging characteristics and workload
3. You think you have a problem with logging

The detailed information is written to ddname //LOG.
Data in CSV format is written to //LOGCSV, see Log statistics in CSV on page 45
The key throughput indicators are displayed in CSV format in Logbusy on page 46

See SCSQMACS(CSQDQJST) for the layout of the SMF record.

Has my queue manager reached the limit at which it can log data?

There are two factors which limit the rate at which MQ can write to the log data sets.

1. Pages per I/O
2. How busy the logging task is.

Pages per I/O

It is more efficient to process many pages per I/O.
You can get many pages per I/O when processing large messages, large units of work, or channels with a large batch size. If you have only small messages (under 4KB) you may only get one or two pages per I/O. Increasing the number of concurrent tasks processing messages may increase this, but the number of pages per I/O may still be low.

On an IBM performance machine we have seen the upper limit of pages per I/O of about 90. This is with 1MB messages and many gets per commit.

How busy the logging task is

Within the queue manager there is a task which does the logging.

This task

1. Issues a write request to each log, this write one or more 4KB pages.
2. Waits for the write request(s) to end
3. Resumes any tasks waiting for I/O in the data just written
4. Waits for the next request to write data. In a busy system there is usually no wait time.
5. Loops to the top.

In a busy system, most of the time spent by this task is waiting for disk I/O.

The time for an I/O request breaks down into two parts.

1. Time to set up the request

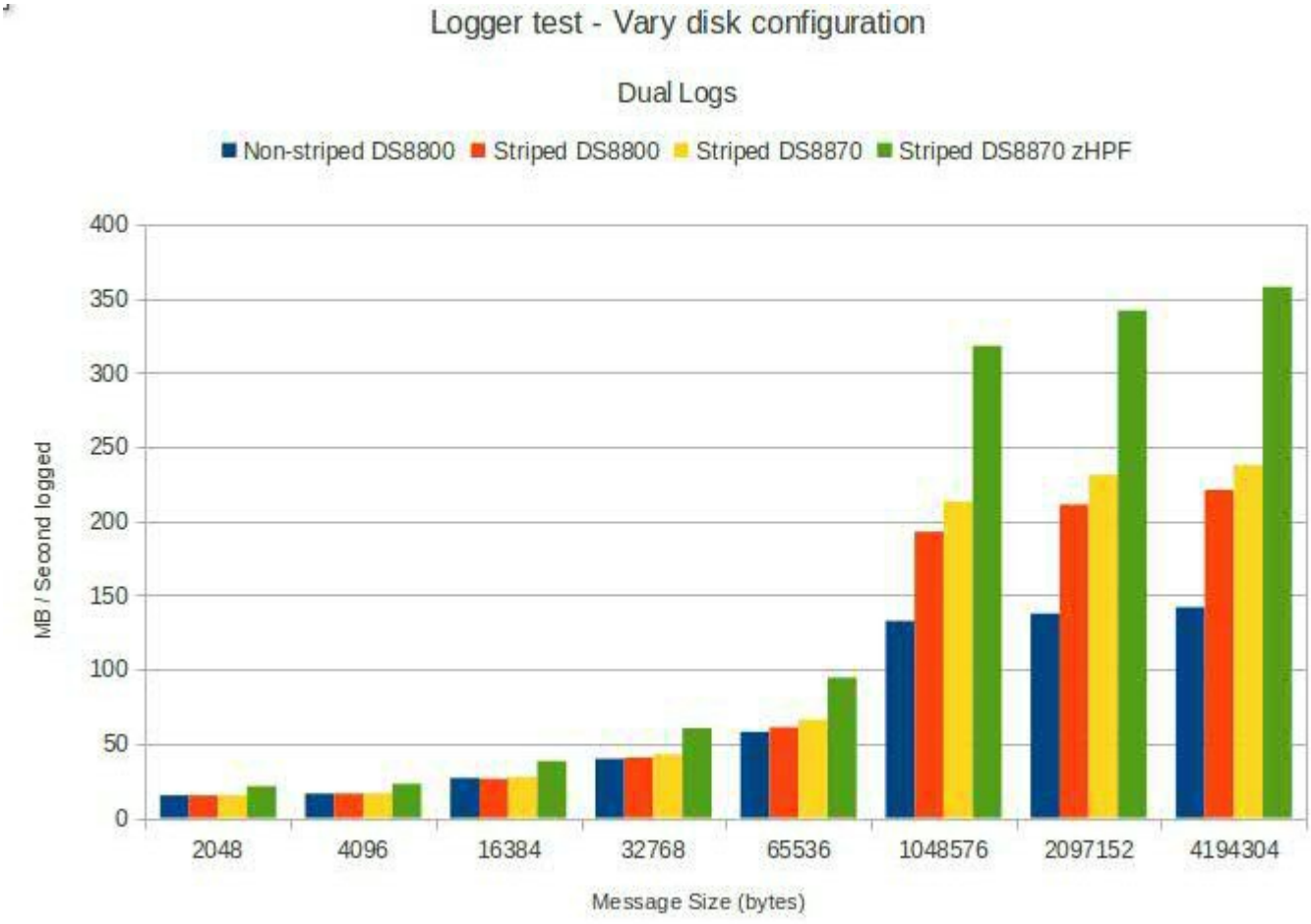
2. Time to transfer the data this varies with the amount of data writen.

As more data is logged, the I/O rate can decrease. More data is written per I/O request, to the duration of the I/O request increases, so there are fewer requests per second. The I/O rate is not so important as how busy the log task is.

At a simple level the task busy is *rate of I/O requests * average time for I/O request.*

This “task busy” is calculated as part of the log statistics displayed below.

To improve throughput you need to reduce the I/O time. The time for an I/O request is very dependant on the workload and the I/O configuration, including the path to the DASD (channels) as well as the DASD subsystem itself. For example on a performance machine in IBM



The chart shows the effect of different hardware configurations using MQ V8. This is the logging rate for single log in a dual log environment.

1. Message sizes from 2KB to 4MB were used.
2. Striping only makes a difference with larger messages
3. Faster disks (DS8870) gave a bit more uplift (at the top end) but little difference with smaller messages (because the internal logger task was 99% busy for the 2KB messages)
4. zHPF gave a big increase especially with large messages.

Using mirrored DASD will decrease throughput depending on the distance.

To see if your queue manager is approaching its logging limit see Logbusy on page 46.

Regular checks

You should review the log statistics regularly to get a profile of the logging statistics by hour and by day. This will allow you to compare a bad day with a good day to see if there is a significant change in the logging rate.

This data is reported in Log statistics in CSV on page 45.

You need to review

1. Wait for buffer. Ideally this should be zero. A value greater than zero means you have run out of buffers. Monitor this for increases to the value.
2. Log task busy.
3. Pages per I/O

Depending on your workload you may have log busy > 90% and be limited when

1. pages per I/O is small. Typically lots of short messages are being processed
2. pages per I/O is large. Typically large messages are being processed, or many concurrent tasks processing short messages

You need to know the profile of pages per I/O for your environment.

You think you may have a problem with logging.

If you think you have a problem with the logging, you may need to look at the detailed records.

Key fields to check. See the text following to identify the specific fields you need to check.

Wait for buffers should always be zero. If this number is greater than zero, the internal buffer filled up and there was no space to store any more data. Applications processing persistent messages will be delayed until buffers are available.
Space in this buffer is freed when the I/O using the space has completed. This problem can be caused by

1. Active logs filling up – perhaps due to a problem with archiving.
2. More data is being logged than the I/O system can handle. Improve the I/O rate, perhaps by striping the logs, or moving the log data sets to low use volumes, or reduce the work on the queue manager.

The number of pages per I/O. If this is small you may have many small transactions. Having more transactions running concurrently should increase the pages per I/O and so the throughput

There have been some unusual disk response times. With MQ V7.1 APAR PM61284 and MQ V800 the log statistics report the average I/O response time, and the maximum response time seen in the interval. This may indicate a problem with the DASD subsystem.
If the I/O response time for writing one page is significantly different from what you expect this may indicate a problem with the I/O subsystem.

Interpreting MP1B output

Records always produced

```
MVCA MQ7A 2013/01/04 14:29:37 VRM:710
   Wait for buffers(should be 0):            0
   Total Number of pages written:         5078
   Total Number of write requests:         172
   Pages written per I/O:                   29
```

Records produced if detail >= 10.

```
MVCA MQ7A 2013/01/04 14:29:37 VRM:710
   Wait for buffers(should be 0):            0
   Total Number of pages written:         5078
   Total Number of write requests:         172
   Pages written per I/O:                   29
   Write_Wait         0, Write_Nowait       391, Write_Force          1, WTB         0
   Read_Stor          0, Read_Active          0, Read_Archive         0, TVC         0
   BSDS_Reqs          3, CIs_Created       2526, BFWR               113, ALR         0
   ALW                0, CIs_Offload          0, Checkpoints          0
   Read delayed       0, Tape Lookahead       0, Lookahead Mount      0
   Write_Susp        13, Write_Reqs         172, CI_Writes         5078
   Write_Serl         0, Write_Thrsh        100, Buff_Pagein          0
```

Records produced at V8 or with APAR PM61284

```
With MQ V7.1 APAR PM61284  and V8 the format is
From 2014/06/06,19:01:00.017286 to 2014/06/06,19:02:00.016951, duration    60 seconds.
 Wait for buffers(should be 0):             0
 Total Number of pages written:       2783396
 Number of pages written/sec:           46389
 Amount of data written/sec:              181 MB/Sec
 Total Number of write requests:       161826
 Number of write requests/sec:           2697
 Pages written per I/O:                    17
 Total number of read requests:             0
 _____,__ write requests,        CIs, Average I/O , After I/O   , pages/IO
                                      time in uSec, time in uSec,
 Log 1, 1 page      45373,      45373,         275,             5,        1
 Log 1,>1 page      35540,    1346300,        1192,             4,       38
 Log 2, 1 page      45373,      45373,         274,            18,        1
 Log 2,>1 page      35540,    1346300,        1174,            50,       38
 Standard deviation of first log, 1 page per I/O, response time +-        3
 Log 1, 1 page Longest I/O      149742 at 2014/06/06,18:01:37.439014 UTC
 Log 1, 1 page Longest Request  149746 at 2014/06/06,18:01:37.439014 UTC
 Log 1,>1 page Longest I/O      331246 at 2014/06/06,18:01:37.095495 UTC
 Log 1,>1 page Longest Request  331249 at 2014/06/06,18:01:37.095495 UTC
 Log 2, 1 page Longest I/O       46341 at 2014/06/06,18:01:15.381098 UTC
 Log 2, 1 page Longest Request  149742 at 2014/06/06,18:01:37.439018 UTC
 Log 2,>1 page Longest I/O       52811 at 2014/06/06,18:01:39.381000 UTC
 Log write rate      92MB/s per copy
 Logger I/O busy : 95.11%
 Logger task busy: 97.99%           (V800)
```

This information shows log 1 had 45373 write requests where it wrote one page per I/O. The

average I/O time of these requests was 275 microsecond.
Log 1 also had 35530 requests to write more than one page – the average was 38 pages per I/O. The average elapsed time for the I/O was 1192 microseconds. Similarly for log 2.
Over time you should expect the average time to write just one page to vary by a small amount. When more than one page is written per I/O, the duration of the I/O will depend on the amount of data written.

When an I/O operation has finished the logging task has to be resumed. The duration between the I/O complete and the time the logging task processed the next instruction is recorded as the '*After I/O time in uSec'* This value should be small. If there is a shortage of CPU, this task may be delayed, so a significant increase in this value may indicate a shortage of CPU.
Typically the time for log 2 is greater than log 1, because it includes time for processing log 1.

The line

```
Standard deviation of first log, 1 page per I/O, response time +-        3
```

reports how consistent the response time was for writing one page. This should be close to 0. If this is larger than your average response time this may indicate your I/O subsystem is not giving you consistent response time.

The time of data when the longest I/O occurred, and the longest time the logging task was suspended is also recorded.
So in the report where it has

```
Log 1, 1 page Longest I/O      149742 at 2014/06/06,18:01:37.439014 UTC
```

The longest I/O time was 149742 microseconds and this occurred at 18:01:37. You can use this time to correlate with any out of line events, such as transactions taking longer than usual.

The log write rate is the number of pages written per log per second * 4096 bytes per page.

The logger I/O busy is the sum of

- Maximum of (time for log 1 to write 1 page, time for log 2 to write 1 page)
- Maximum of (time for log 1 to write >1 page, time for log 2 to write >1 page)

So the value of 95.11% shows that there was I/O active for most of the time. The queue manager is close the the limit of what it can log. To improve throughput you will need to improve the I/O response time to the DASD.

The response time for writing one page should be consistent over time. The response time when multiple pages are written will depend on the number of pages written and whether the log datasets are striped. So monitoring the response time of writing one page gives you a good indication of the behaviour of your DASD over time.

The *logger task busy* field is new in V8 and shows how busy the log task was. The value in the SMF record is the time the log task was idle. So the time the log task was active is (duration – qjstslptu).

The example data shows that the queue manager is close to the limit at what it can log.

This can be seen from

- The majority of the requests processed multiple pages `1346300 (> 1 page per I/O) compared to 45373(1 page per I/O)`
- `High number of pages per I/O (38)`
- `Logger I/O % busy (95%.11)`
- **`Logger task busy: 97.99`**

With 10 batch jobs putting and getting 1KB persistent messages the key statistics were

```
_____,__ write requests,        CIs, Average I/O , After I/O   , pages/IO
                                     time in uSec, time in uSec,
Log 1, 1 page     209425,     209425,         218,             3,        1
Log 1,>1 page          1,          2,        2078,             2,        2
Log 2, 1 page     209425,     209425,         221,             7,        1
Log 2,>1 page          1,          2,        2066,             3,        2
Log write rate      13MB/s per copy
Logger task busy: 82.53%
Logger I/O busy : 79.86%
```

This shows the logging task was busy, but the majority of the I/O had just one page per I/O.
The queue manager can log more data, but more concurrent jobs will be needed (or more data written before the commit request).

Log statistics in CSV

The log statistics give information about the amount of data written to the log datasets.

The data reported is for one log, so if you have dual logs you will have similar data for the second logs.

The records are written to ddname //LOGCSV.

'z/OS','QM ','Date ','Time','MB Written','MB/SEC','MB Used','Pages per I/O','Checkpoints',

MVCA,MQ7A,2013/02/09,16:50:00,51749, 172,25675, 31, 1
MVCA,MQ7A,2013/02/09,16:55:00,48389, 161,24009, 32, 1

Where

z/OS	Is the name of the z/OS image
QM	Is the queue manager name
Date	Date in YYYY/MM/DD formatting
Time	The time in HH:MM:DD format
MB Written	This is field (qjstioci for 1 page and > 1 page for the first log)converted to MB. This field is the number of pages written to DASD. A page may be written multiple times if it was not full, and a commit was issued
MB/SEC	This is field *MB Written/ SMF duration*
MB Used	This is the amount of log data used qjstbffl in MB. This can be smaller that the MB written, as a page may get written multiple times
Pages per I/O	As rate of data written to the increases, then this value increases. The maximum value depends on your environment, especially the rate at which your can write data to your DASD.
Checkpoints	The number of checkpoints that occurred in the interval.

Logbusy

With WMQ V7.1 APAR PM61284 and WMQ V8 information is provided on log data set I/O response times as seen by the logger.

The duration of the interval is in qwhsdurn.

Information is reported when one page is written, and when multiple pages are written, and for each log.

Qjstiototsus is the total time spent suspended for I/O.

The busy time is calculated as

max1page time = max(qjstiototsus for 1 page log1,qjstiototsus for 1 page log 2)

maxNpagetime = max(qjstiototsus for >1 pages log1,qjstiototsus for > 1 pages, log2)

busy time = max1page time + maxNpagetime

100 * Busy time/ duration is the logBusy% time.

The logbusy data set has the following data

z/OS,QM,Date,Time,logBusy%,MB/Sec, pages/IO

z/OS	Is the name of the z/OS image
QM	Is the queue manager name
Date	Date in YYYY/MM/DD formatting
'Time	The time in HH:MM:DD format
logBusy%	See logBusy% above
MB/Sec	This is the same as in Log statistics in CSV above
Pages/IO	This is the overall average pages/IO.

Additional messages produced in MESSAGE output file.

Message: MQQJST00I ... QJST read log buffers from storage n > 0

Detail: 15
When: If qjstrbuf > ReadLogBuffers
Reason: The field Read_stor (QJSTRBUF) is the number of requests which were satisfied from the log buffers, and did not require the log datasets to be read. The value of this field is greater than the MQSMF parameter ReadLogBuffers.

Reading data from the log can often occur, for example when an application processing persistent messages rolls back the work. If the amount of roll back activity is large, then this may indicate a problem in the application or the environment.

Action:
Monitor the number of log buffers read from storage and specify a suitable value in the ReadLogBuffers parameter.

Message:MQQJST01W ... QJST read log buffers from active logs n > 0

Detail: 10
When: qjstract > 0
Reason: Message MQQJST00I is for the number of log buffers read from memory. If the data is not in memory, then the data is read from a log data set.
This message is produced when data was read from an active log dataset.
In normal operation you do not expect messages to be read from the active log datasets.
Action:
Investigate why data was being read from the active log datasets. For example, this could be due to transactions processing lots of data in an unit of work, or taking a long elapsed time before work is committed; and then rolling back.

Message: MQQJST02S ... QJST read log buffers from archive logs n > 0

Detail:4
When: qjstrarh > 0
Reason: This message is produced if log data is read from an archive log dataset, which should not happen.
Action:
See MQQJST01W on page 47.
You may need to increase the number of active log datasets. Check the log datasets are large enough.

Message: MQQJST03E ... QJST Number of checkpoints n > m

Detail:10
When: qjstllcp > 10
Reason: This message is produced when there are more checkpoints in the SMF interval than expected.
Action:
This may indicate a peak in activity.
The active logs may be too small.
Monitor the qjstllcp and specify a suitable value in the parameters

Message: MQQJST04E ... QJST Number of buffer paged in n > 0

Detail:10
When: qjstbpag > 0
Reason: Some log requests were delayed because the log buffers had to be paged in. This value should be 0
Action: This indicates that there may be a real storage shortage. Investigate and resolve this.

Message: MQQJST05E ... QJST Number of read accesses delayed n

Detail:10
When: qjstwur > 0
Reason: Some log read requests were delayed due to an archive log not being available. This field should be 0
Action: Investigate why the archive log was being used.

Message: MQQJST06E ... QJST Number of look ahead tape mounts attempted n

Detail:10
When qjstlama > 0
Reason: An archive log on tape was needed.
Action: Investigate why the archive log was being used.

Message: MQQJST07E ... QJST Number of look ahead tape mounts performed n > 0

Detail: 10
When: qjstlams > 0
Reason: An archive log on tape was needed.
Action: Investigate why the archive log was being used.

Message: MQQJST07E ... QJST Number of reads delayed for tape contention n

Detail: 10
Reason: There was contention for an archive log on tape
Action: Investigate why the archive log was being used.

Message: MQJST09W … QJST % requests waiting for buffer n > m

Detail:15
When: pcwtb >= 1
Reason: There were no log buffers available to be used, they were either being written to the active log datasets, or waiting to be written to the active log datasets.
This can be caused by

- increased application traffic,
- writes to the active log datasets are slower usual (this may be a hardware related problem)
- or all of your active logs are full.

Action: Investigate your active log DASD and check this is not being constrained.
In the long term you may need to move work to a different queue manager, as you are reaching a limit to the rate at which you can log data.
Monitor how frequently this message occurs. If it only occurs infrequently, this may be acceptable. If it starts to occur more frequently then you need to take action to improve the rate at which you can log data (stripe the datasets, or move the logs to faster DASD) or try to reduce the amount of data logged by this queue manager.

Message: MQQJST10E … QJST High logging rate n > m MB/Sec

Message: MQQJST11W … QJST OK logging rate n > m MB/Sec

Message: MQQJST11I … QJST logging rate is low n < m MB/Sec

Detail:15
When: the number of CIs written (qjstciwr) in the specified time exceeds the parameter value
Reason: The number of 4K pages written to the log is qjstciwr, This value in MB is qjstciwr * (4*1024)/(1024*1024=1MB) = qjstciwr * 4/1024 MB
The SMFTime parameter is the time you have specified between SMF records, so the rate at which data is logged is (qjstciwr * 4/1024 MB)/SMFTime*60 (in seconds).

If this data rate exceeds the HighLogRateMB parameter, then message
MQQJST10E … QJST High logging rate n > m MB/Sec is produced
else if this data rate exceeds the OKLogRateMB parameter, then message
MQQJST11W ...QJST OK logging rate n > m MB/Sec is produced
else message
MQQJST11I ...QJST logging rate is low n < m MB/Sec is produced.
You can use this to monitor your logging rate.

Action: Monitor your logging rate and select a high and OK values.
If you find the logging rate is often larger than your HighLogRateMB value, then this may indicate an increasing amount of data being logged. You should investigate moving your logs to faster DASD, striping the logs, or moving work from the queue manager.

Message MQQJST16W … QJST Log write task nn% busy

Reason: The internal logger task was more than 50% busy
Action: Investigate to see if the I/O response time can be improved.
Action: This gives you early warning that the logger is busy. If the busy time continues to increase you should consider moving the log data sets to faster DASD, or reducing the persistent workload on the queue manager.

Message MQQJST17S … QJST Log write task very busy at nn % busy

Reason: The internal logger task was more than 50% busy
Action: This internal logger task is very busy and may be close to the thoughput limit.

Message MQQJST18S … QJST % of interval spent doing disk I/O nn %

Reason: The percentage of the duration there was I/O active was more than 90% busy
Action: Investigate to see if the I/O response time can be improved.

Message MQQJST19W … QJST % of interval spent doing disk I/O nn %

Reason: The percentage of the duration there was I/O active was more than 50% busy
Action: This gives you early warning that the queue manager is spending a significant amount of time doing log I/O. If the busy time continues to increase you should consider moving the log data sets to faster DASD, or reducing the persistent workload on the queue manager.

Buffer pool Statistics

The buffer pool statistics give information about the buffer pools.
In version 8 buffer pools can be in 64 bit storage and can be page fixed.

The records are written to ddname //BUFF.

See SCSQMACS(CSQDQPST) for the layout of the SMF record.

Records always produced.

```
MVCA MQ7A 2013/01/04 14:29:37 VRM:710
= BPool   4, Size 50000,%full now  1, Highest %full  1, Disk reads     0
```

Where the fields are

`BPool` The buffer pool number

`Size` Number of buffers in this buffer pool when the SMF record was created

`%full now` How full the buffer pool was when the SMF record was created. This value is the number of buffers in the buffpool – number of free buffers.

`Highest %full` This is the maximum usage of the buffer pool during the period, as a percentage of the total size of the buffer pool.

`Disk reads` The number of pages read from disk. You get best performance if all of your data is in the buffer pool. Reading from the pageset usually indicates the buffer pool had filled, and so buffers had to be written out to the page set

record produced if detail >= 5

```
  = BPool   4, Size 50000,%full now  1, Highest %full  1, Disk reads     0
  > 04 Buffs    50000  Low    49470  Now    49473  Getp  3962281  Getn  3481517
    04 Rio          0  STW  3985069  TPW    25410  WIO      7053  IMW       886
    04 DWT          0  DMC        0  STL        0  STLA        0  SOS         0
    04 Above the bar   PAGECLAS 4KB
```

Where `04 Above the bar  PAGECLAS 4KB` is new in V800.

Buffer pool Statistics CSV

```
'MVS','QM','Date','Time','BP','size','lowest free',
SOS,DMC,DWT,'# get new pg','# get old pg','# read I/Os',
'# pg writes','# write I/Os','# sync write','LOCATION','DATACLAS'
```

MVCA,MQPG,2013/02/11,08:00:00, 1,10000, 9980,   0

```
MVCA,MQPG,2013/02/11,08:00:00,  2, 1000,  996,    0
MVCA,MQPG,2013/02/11,08:00:00,  3, 1000,  994,    0
MVCA,MQPG,2013/02/11,08:00:00,  4,50000,49995,    0
```

Subsystem statistics

The subsystem statistics give information on how many jobs ended normally, and how many jobs ended abnormally.

The records are written to ddname //EOJ.

Records always produced.

```
MVS  QMGR Date       Time     VRM          Jobs EOT     Jobs EOM
MVCA MQPA 2013/01/06 11:36:46 VRM:701            10            0
MVCA MQPA 2013/01/06 11:37:31 VRM:701             2            0
```

Jobs EOT is the count of TCBs that ended normally. Jobs EOM is the number of jobs that ended abnormally.

Message manager

The message manager statistics gives information on how many API requests there were in the interval.

The records are written to ddname //MSGM.

See SCSQMACS(CSQDQMST) for the layout of the SMF record.

```
MVCA MQ7A 2013/02/09 01:00:00 VRM:710 : no data found
MVCA MQP0 2013/02/09 01:00:00 VRM:710
   MQOPENs         0,  MQCLOSEs         0,  MQGETs         6,  MQPUTs         6
   MQPUT1s         0,  MQINQs           0,  MQSETs         0,  C ALL H        0
   MQSUBs          0,  MQSUBRQs         0,  MQCBs          0
   MQCTLs          0,  MQSTATs          0,  Publish        0
```

If all the value are zero, *no data found* is displayed.

The field names are:

MQOPENs	MQOPEN
MQCLOSEs	MQCLOSE
MQGETs	MQGET
MQPUTs	MQPUT
MQPUT1s	MQPUT1
MQINQs	MQINQ
MQSETs	MQSET
C ALL H	Close all handles – issued at the end of the task
MQSUBs	MQSUB
MQSUBRQs	MQ Subscription request
MQCBs	MQ call Back
MQCTLs	MQ control call back
MQSTATs	MQ retrieve status information
Publish	The total number of messages published.

Message manager CSV

The message manager statistics give information on how many API requests there were in the period.

The records are written to ddname //MSGMCSV with one line per record, suitable for importing into a spread sheet.

```
'MVS','QM','   Date','  Time','    Puts','  Put1s','      Gets
MVCA,MQP0,2013/02/09,01:00:00,         0,         0,         6
MVCA,MQPG,2013/02/09,01:00:00,         0,         0,       326
```

Topics

MVCA MQ7A 2013/02/12 18:02:35 VRM:710
Total Subs 2 Durable Subs 0 Expired Subs 0
Total messages 6 Single publish 0
API sub HW 0 Sub LW 0 Tot Pub 6
ADMIN :Sub HW 0 SUB LW 0 Tot Pub 0
PROXY :Sub HW 0 SUB LW 0 Tot Pub 0
Single PUB HW: 2 Pub LW 2 Pub Nosub 0
Max Pub time 259 Avg pub time 76

The Topic section gives information on Publish Subscribe

The records are written to ddname //TOPIC.

See QTST in SCSQC370(CSQDSMFC)) for the layout of the SMF record.

Data manager

The data manager manages the links between messages and queues. It calls the buffer manager to process pages with messages on them.

The records are written to ddname //DATA.

See SCSQMACS(CSQDIST) for the layout of the SMF record.

```
MVCA MQ7A 2013/02/09 10:20:00 VRM:710
   Obj Cre         1, Obj Puts        0,  Obj Dels        0,  Obj Gets         3
   Locates         4, Stgclass        0,  Enum           25
   Msg Gets     1000, Msg Puts     1000
   Lock MM         0, Rel MM         32,  Delete MM         0
   Read Ahead:IO  29,:Buffer         22,  Gets disk      17,  Gets BP          3
```

The fields names are

Obj Cre The number of object creates – such as DEFINE QLOCAL
Obj Puts The number of times an object was changed, such as ALTER

	QL(Z)DESCR(Comment)
Obj Dels	The number of times an object was deleted
Obj Gets	The number of times an object definition was got, for example for a display, or for an alter
Locates	The number of times a locate object was issued. This could be for displaying or altering an object, or application activity opening a queue.
Stgclass	The number of requests to alter a STGCLASS
Enum	The number of requests to find an object
Msg Gets	The number of MQGET requests
Msg Puts	The number of MQPUT or MQPUT1 requests
Lock MM	The number of Lock Marked Message requests
Rel MM	The number of Release Marked Message requests
Delete MM	The number of Delete Marked Message requests

If messages are being got in order, and the messages are being read from the page set, then an internal task may be started to perform read ahead to read messages from the disk, so that they are in the buffer pool when the application next does an MQGET.

The read ahead data for messages in MQ V6, and in all releases for getting objects from page set 0.

Read Ahead:IO The read ahead task got pages which required I/O to a page set

Buffer	The read ahead got a page – but it was already in the buffer pool and did not require an I/O to the page set
Gets:disk	An MQGet was suitable for a read ahead. This is the count of pages the application got which required I/O to a page set
Gets BP	An MQGet was suitable for a read ahead. This is the count of pages the application got which were already in the buffer pool.

```
Read Ahead:IO   0,:Buffer          3,  Gets disk       1,  Gets BP       2660
```

Lock manager

The queue manager uses locks to prevent concurrent updates to resources.

This information is reported in ddname //LOCK. It is usually useful only to IBM.
See SCSQMACS(CSQDQLST) for the layout of the SMF record.

```
                                                  Gets Already Held      Releases
MVCA MQPA 2013/01/06 09:00:00 VRM:701             6673            0          1934
MVCA MQPA 2013/01/06 09:15:00 VRM:701             4566            0          1359
```

Storage manager

The storage manager is responsible for managing virtual storage within the queue manager.

This information is reported in ddname //STG. It is usually useful only to IBM.

See SCSQMACS(CSQDQSST) for the layout of the SMF record.

```
MVCA MQPA 2013/01/06 09:00:00 VRM:701
   Fixed pools   : Created         0,  Deallocated         0
   Fixed segments: Freed           0,  Expanded            2,  Contracted         1
   Varbl pools   : Created         0,  Deallocated         0
   Varbl segments: Freed           0,  Expanded            6,  Contracted         5
                   Getmains        1,  Freemains           1,  Non-zero RCs       0
                   SOS             0,  Contractions        0,  Abends             0
```

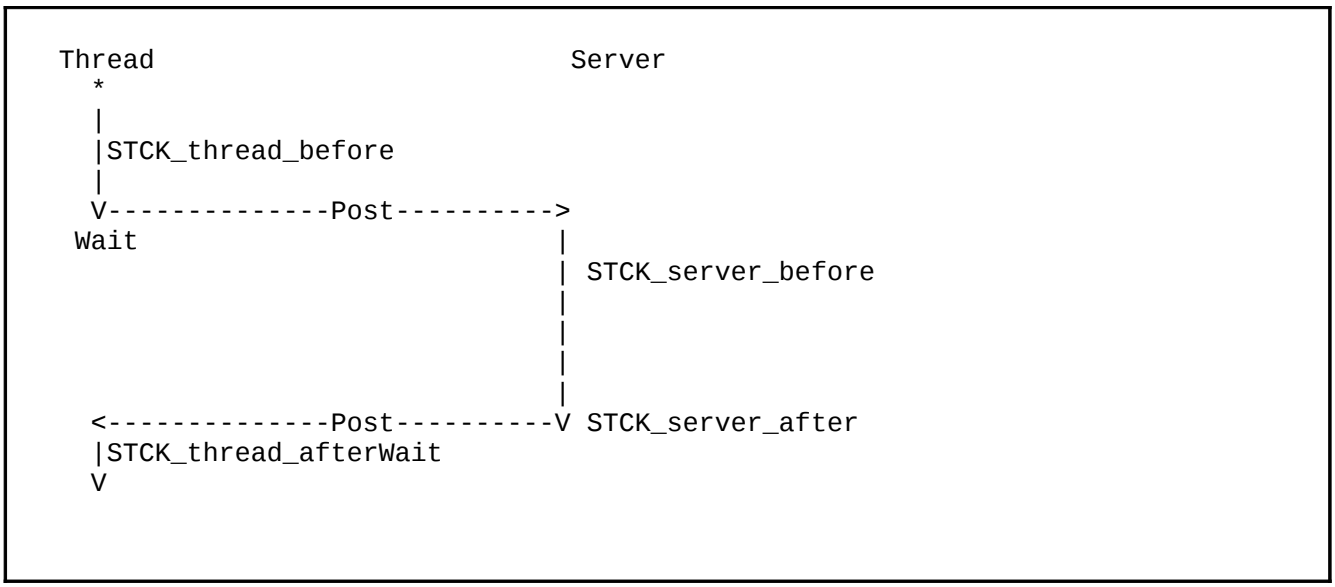
If Contraction (QSSTCRIT) is non zero, this indicates a severe problems with lack of storage within the queue manager.
If SOS (QSSTCONT) is non zero this indicates a problem was detected and recover actions were taken. This indicates a severe problem within the queue manager.

DB2 statistics

The DB2 statistics gives information about the DB2 tasks running in the queue manager.

The information is reported in ddname //DB2.
See SCSQMACS(CSQD5JST) for the layout of the SMF record.

The DB2 manager manages the interface with the DB2 database that is used as the shared repository.
When using shared queues, object definitions and other information are stored in DB2 tables.
DB2 requests are made from the queue manager by passing a request to a pool of server tasks that issue the DB2 request on behalf of the applications.
The figure below shows how DB2 requests are issued
Flow of a request for a DB2 service from a thread to server task



The processing for a thread wanting to issue a read request is as follows:

1. The thread puts a request onto a server work list.
2. The thread determines the current time (`STCK_thread_before`).
3. The thread posts a server task.
4. The thread waits.
5. The server task wakes up, and determines the current time (`STCK_server_before`).
6. The server takes the first request off the server work list and issues the DB2 request.
7. When the request has ended it posts the thread task.
8. The server task determines the current time (`STCK_server_after`) and updates the statistics:
    1. It increments the number of read requests `READCNT`.
    2. It calculates the time taken it took to process the request, `STCK_server_after-STCK_server_before` and adds this to the cumulative time `READSCUW`.
    3. If the time for the request was larger than the previous maximum if replaces the

READSMXW with the delta.

Note: For other request, other counters are updated. These are `LIST*`, `UPDT*`, `DELE*`, and `WRIT*`.

9. The original thread wakes up and determines the current time (`STCK_thread_after`) and updates the statistics:
    1. It calculates the time spent waiting (`STCK_thread_after - STCK_thread_before`) and adds this to the cumulative time `READTCUW`.
    2. If the time spent waiting for the request was greater than the previous maximum it replaces the `READTMXW` with the larger value.

Note: For other request, other counters are updated. These are `LIST*`, `UPDT*`, `DELE*`, and `WRIT*`.

10. The thread continues processing.

The processing is similar for update, write, and delete requests. The list request is more complex and can result in reads being done from the server task issuing the list request.

Shared-channel-status and shared-sync-key tables

If you are using shared channels, shared-channel-status information and information about the shared-sync-queue are stored in DB2 tables. The fields with names starting SCS* are for DB2 selects, inserts, updates, and deletes from the shared-channel-status table. The fields with names starting SSK* are for DB2 selects, inserts, updates, and deletes for information about the shared-synch-key table.

The shared-sync-key table is used to locate the message id for messages on the shared sync queue. The Shared Channel Sync queue is used when the channel NPMSPEED(NORMAL) is used. Messages on the queue have information about the status of messages in a batch. The Shared Sync Key table, provides a mapping from channel name, XMITQ name, and remote queue manager name to the messages for the channel in the Shared Channel Sync queue.
Information is inserted into the Shared Sync Key table, when a channel processes messages with NPMSPEED(normal) for the first time. Both of these have times for the thread and the server, as described above.

```
MVCA MQPA 2013/01/15 16:30:00 VRM:701
 Tasks  : Servers    8, Active    9, Conns       0, Discs       0
          HighMax    3, Abend       0,  Requeue       0
               Count  Task avg   Task max   DB2 avg  DB2 max(ms)  (Task-DB2)Avg    Max
 List       :    180         2         6         2         5                0       0
 Read       :      1         3         3         3         3                0       0
 SCS Select :     15         2         4         2         4                0       0
```

The first column is the request type. These can be

List	Interpretation
List	This is when a query is done to the DB2 database, for example as a result of a display command
Read	Read of the definition of an object stored in DB2
Update	Update of the definition of an object stored in DB2
Write	Insert the definition of an object stored in DB2

List	Interpretation
SCS Select	Shared-channel-status table Select
SCS Insert	Shared-channel-status table Insert
SCS Update	Shared-channel-status table Update
SCS Delete	Shared-channel-status table Delete
SSK Select	Shared-synch-key Select
SSK Insert	Shared-synch-key Insert
SSK delete	Shared-synch-key Delete
Blob Select	Shared large message Select
Blob Insert	Shared large message Insert
Blob Update	Shared large message Update
Blob Delete	Shared large message Delete
Blob List	Shared large message List

MQQ5ST01S ... Q5ST Abend count n > 0

When: abndcnt > 0
Reason: The DB2 servers tasks have abended.
Action: There will be messages on the job log about why the server task abended. Review these and resolve the problem.

MQQ5ST02E ...Q5ST Retry count n > 0

When: requcnt > 0
Reason: A DB2 server task failed and the request was retried.
Action: There will be messages on the job log about why the server task abended. Review these and resolve the problem.

MQQ5ST10S … Q5ST Number of deadlock conditions n

When: deadcnt > 0
Reason: Some deadlocks occurred in DB2, and a request was rolled back.
Action: If this happens frequently contact your IBM service representative.

MQQ5ST04W … Q5ST DB2 Average read time n> m

When: The number of DB2 requests is greater than 10, and readscuw/readcnt > DB2Time parameter
Reason: The average DB2 time for a read (n) is greater than the DB2 time passed as a parameter(m).
Action: Investigate any DB2 delays. You may need to set DB2Time to a more suitable value.

Accounting data describing the task

Information is displayed about the task.

For summary information see
1. TaskElapsed Time on page 72
2. TaskCSV on page 73.

See wtid in SCSQC370(CSQDSMFC) for the layout of the task ID information, wtas in SCSQC370(CSQDSMFC) for the task layout, and wq in SCSQC370(CSQDSMFC) for the queue specific layout.

For batch job:

```
MQ06 Batch Jobname:PAICECC  Userid:PAICE
```
This shows the data is for batch job PAICECC and userid PAICE and queue manager name MQ06.

For CICS transaction:
```
MQ01 CICS IYFFC001 opid:PAICE    userid:SCENSTC  Tran:CN15 task:0001664c
```
This shows the data is for CICS region IYFFC001, terminal userid PAICE CICS region userid SCENSTC, transaction name CN15, CICS task number 1664c on queue manager name MQ01.

For a channel:
```
MQ01 MOVER Jobname:MQ01CHIN Userid:POC005
Channel COLIN61A             ::ffff:9.20.5.21
```
This shows the data is for the mover MQ01CHIN with userid POC005 on queue manager name MQ01.
The channel name was COLIN61A with IP address ::ffff.9.20.5.21

Detailed information common to all types of applications

```
Start time Feb  1 13:24:13 2013 Started in a different time interval
Interval   Feb  1 13:27:13 2013 - Feb  1 13:27:13 2013 : 112.913245 seconds
Other reqs : Count               119
Other reqs : Avg elapsed time     36 uS
Other reqs : Avg CPU              22 uS
Other reqs : Total ET              0.004345 Seconds
Other reqs : Total CPU             0.002628 Seconds
> Latch 11, Total wait          0 uS, Waits          1, Name DMCSEGAL|SSSCONN
> Latch 12, Total wait      37806 uS, Waits         13, Name DMCNMSPC|XMCHASH
> Latch 16, Total wait       8113 uS, Waits        182, Name BMXL2   |RMCRMST |
RLMARQC
> Latch 19, Total wait      74869 uS, Waits        196, Name BMXL3   |CFXML2  |
SRH1_L19
> Latch 21, Total wait       6661 uS, Waits       2045, Name RLMLWRT
> Latch 32, Total wait          0 uS, Waits          2, Name SMCPHB
Longest latch wait at 0000000020b48598 74869 uS
Avg Latch time per UOW          2770 uS
Commit count                      44
Commit avg elapsed time         6061 uS
Commit avg CPU time               13 uS
Backout count                      2
Backout avg elapsed time      260912 uS
Backout avg CPU time          260912 uS
Log write count                19312
```

```
Log write avg et                  765 uS
Log I/O Bytes              1015644601
Log Force Count                 19312
Log Force Avg elapsed time        765 uS
Suspend Count                      46
Suspend Avg elapsed time        17129 uS
Total suspend time                  0.787944 Seconds
Pages old                       46146
Pages new                     1275923
…
Log wait/Commit                 1023 uS
  Log force/Commit                1023 uS
  Grand total CPU time             117 uS
  Grand Elapsed time              4464 uS
```

Interpretation

Task related

The value in () at the start of the line is the minimum Detail level required to display the record.

`( 1)  Grand total CPU time              n uS`
This is the sum of all the CPU time for all MQ requests, used by the task.

`( 1)  Grand Elapsed time                n uS`
This is the sum of all of the elapsed times or all MQ requests, used by the task.

`( 3) Start time s Started this interval`
s is the start time of the transaction in the format *Month day hh:mm:ss yyyy*. If it is in the current SMF period, then is also displays *Started this interval*. If it started in a different SMF interval it displays *Started in a different time interval.*

`( 3) Interval   ss - se: n seconds`
ss is the start time of this record, se is the end time of the interval. The format is Month day hh:mm:ss yyyy. N is the number of seconds in the interval.

`(15) Avg Latch time per UOW             n uS`
This is the total latch wait time / (count of commits + count of backouts).

`( 5) Commit count                       n`
This is the number of commits.

`( 5) Commit avg elapsed time            n uS`
This is the average time per commit. This is the elapsed total time for commits/count of commits.

`( 5) Commit avg CPU time                n uS`
This is the average CPU on the application thread for the CPU time. This is the application CPU time for commits/count of commits. Note, most of the commit activity is done on a SRB running in the queue manager address space, and the CPU used by this SRB is not recorded in this thread value.

`( 5) Backout count                     n`
This is the number of back out requests.

(5) Backout avg elapsed time n uS

This is the average time per backout. This is the elapsed total time for backout/count of backout requests.

(5) Backout avg CPU time %12.1u uS

This is the average CPU on the application thread for the CPU time. This is the application CPU time for backouts/count of backouts. Note,most of the backout activity is done on a SRB running in the queue manager address space, and the CPU used by this SRB is not recorded in this thread value.

(10) Log write count n

This is the number of writes to the log

(10) Log write avg et n uS

This is the average time to write to the log. This is the total write time/count of log requests. This is the total logging including the logging reported under PUT,PUT1, GET and SET.

(10) Log I/O Bytes n

This is the number of bytes logged by the application. This is the total logging including the logging reported under PUT,PUT1, GET and SET.

(10) Log Force Count n
(10) Log Force Avg elapsed time n uS

(10) Suspend Count n

This is the number of times an MQ request was suspended within the queue manager. This may be for internal latches (used to serialise activity), waiting for logging or waiting for other activity.

(10) Suspend Avg elapsed time n uS

This is the total suspend time/count of suspends.

(10) Total suspend time n.nnnnnn Seconds

This is the total suspend time.

(10) Pages old n

This is the total number of requests to get a page from a buffer pool where the contents were needed. Put and gets both increment this number.

(10)Pages new n

This is the total number of requests to get a page from a buffer pool when a new page was needed. Only put requests increment this number.

(10) SMDS : Blocks written n, blocks read n

This is the number of SMDS blocks written or read for all of the queues.

(10) SMDS : Pages written n, pages read n

This is the number of SMDS blocks written or read for all of the queues.

(10) SMDS : Pages read from cache n

When a get request was issued, the data was in the cache and did not need to be read in.

(10) SMDS : Total wait time for I/O n.n Seconds

(10) SMDS : I/O wait time per block n.n Seconds

(10) SMDS : I/O wait time per page n.n Seconds

(4) == SRB CPU time used n.n Seconds

When using publish/subscribe some work is run on an SRB task on behalf of the application. This is the total amount of SRB CPU time used.

(6) Other reqs : Count n

MQ Requests, MQOPEN, MQCLOSE, MQPUT, MQPUT1, MQGET have their own sections in the Queue accounting record. This field is the count of any other requests – including internal request, and for any requests to queues which cannot be recorded against a queue.

(6) Other reqs : Avg elapsed time n uS

This is the total elapsed time for other/count of other requests.

(6) Other reqs : Avg CPU n uS

This is the total CPU time for other/count of other requests.

(5) Other reqs : Total ET n.n Seconds

This is the total elapsed time for other requests

(5) Other reqs : Total CPU n.n Seconds

This is the total CPU time for other requests.

(4) == DB2 activity : n requests

This is the number of requests to DB2 as part of offloading shared queue messages to DB2

(6) > Average time per DB2 request-Server : n uS

This is the average time of the requests to DB2 as part of offloading shared queue messages to DB2, as seen by the server.

(6) > Average time per DB2 request-Thread : n uS

This is the average time of the requests to DB2 as part of offloading shared queue messages to DB2, as seen by the thread. If this is significantly larger than the average time as seen by the server, then there has been some delay due to a shortage of DB2 threads within the queue manager.

(6) > Maximum time per DB2 request-Server : n uS

(6) > Maximum time per DB2 request-Thread : n uS

(6) > Bytes put to DB2 : n

This is the number of bytes written to DB2 when offloading large shared queue messages to DB2.

(6) > Bytes read from DB2 : n

This is the number of bytes read from DB2 when getting shared queue messages that have been offloaded to DB2.

(6) == CF activity : Requests - Single n, Multiple m

When using shared queues there are requests which update one entry in the coupling facility. There are other requests which update multiple entries within the CF – for example a commit with several message in the unit of work.

So the length of time for the update of a single entry should be consistent, but the time to update multiple entries will depend on the number of entries being updates.

(6) > Retries - Single %10.1u, Multiple %

If the CF was 'busy' then a request to the CF may need to be retried. This is a count of the number of retries.

(6) > Average time per single requests : n uS

This is a measure of the CF response time when updating single entries. It is the total time

spent doing IXLLSTE requests/count of IXLLSTE requests.

(6) > Average time per multiple requests : n uS

This is a measure of the CF response time. when updating multiple entries. It is the total time spent doing IXLLSTM requests/count of IXLLSTM requests. This time will depend on the number of entries updated.

(10) == Page set 0 activity : Count n, Avg elapsed n

This is the number of puts and gets which mapped to page set 0.

(15) > Latch n Total wait n uS, Waits n, Name sss

This is information useful to IBM about the latches used within the queue manager to serialise work.

(15) Longest latch wait at xxx %lld uS

This is information useful to IBM about the latches used within the queue manager to serialise work.

Accounting data for a task using local queues

Example data for put and get to a local queue.

```
  Open name                                CP0000
  Queue indexed by MSG_ID                  CP0000
  First Opened   Jan  8 13:11:36 2013      CP0000
  Last Closed    Jan  8 13:11:36 2013      CP0000
  Page set ID                     2        CP0000
  Buffer pool                     2        CP0000
  Current opens                   0        CP0000
  Total requests                  6        CP0000
  Open Count                      2        CP0000
  Open Avg elapsed time          10 uS     CP0000
  Open Avg CPU time              10 uS     CP0000
  Close count                     2        CP0000
  Close avg elapsed time          3 uS     CP0000
  Close avg CPU time              3 uS     CP0000
  Get count                       1        CP0000
  Get avg elapsed time           29 uS     CP0000
  Get avg CPU time               29 uS     CP0000
  Get skipped message count       2        CP0000
  Get TOQ average              3103 uS     CP0000
  Get TOQ maximum              3103 uS     CP0000
  Get TOQ minimum              3103 uS     CP0000
  Get valid count                 1        CP0000
  Get size maximum             1000 bytes  CP0000
  Get size minimum             1000 bytes  CP0000
  Get size average             1000 bytes  CP0000
  Get Dest-Specific               1        CP0000
  Get Persistent count            1        CP0000
  Put count                       1        CP0000
  Put avg elapsed time           33 uS     CP0000
  Put avg CPU time               33 uS     CP0000
  Put + put1 valid count          1        CP0000
  Put size maximum             1000       CP0000
  Put size minimum             1000       CP0000
  Put size average             1000       CP0000
  Put num persistent              1        CP0000
  Curdepth maximum                3        CP0000
  Total Q CPU used               92 us     CP0000
  Total Queue elapsed time       92 us     CP0000
```

Interpretation of the data

The value in () is the minimum value of Detail to display the data.

Opening a queue

*(*) Open name queueName*

This is the name of the queue opened by the application.

*(*) Base name queueName*

This is the name of the base queue if different to the opened name – for example an alias queue.

(10) *Queue indexed by MSG_ID queueName*

How the queue is indexed.

(10) *First Opened Jan 8 13:11:36 2013 queueName*

(10) *Last Closed Jan 8 13:11:36 2013 queueName*

(5) *Page set ID n queueName*

(5) *Buffer pool n queueName*

(14) *Current opens n queueName*

(14) *Total requests n queueName*

(14) *Generated messages n queueName*

How many trigger or event messages were generated. The cost of producing these message will be reported by this queues.

(10) *Open Count n queueName*

How many times this queue was opened by this application.

(10) *Open Avg elapsed time n uS queueName*

The average elapsed time to open the queue. This is calculated as total open elapsed time/count of open requests.

(10) Open Avg CPU time n uS queueName

The average CPU time to open the queue. This is calculated as total CPU time for open/count of open requests.

(10) Open avg topic srb time n uS queueName

When a topic is opened it may attach an SRB to do some processing. This is the average SRB time per open request. It is calculated at the total SRB for open/count of open requests.

Closing a queue

(14) Close count n queueName

(14) Close avg elapsed time n uS queueName

The average elapsed time to close the queue. This is calculated as total close elapsed time/count of close requests.

(14) Close avg CPU time *n uS* *queueName*

The average CPU time to close the queue. This is calculated as total CPU time for close/count of close requests.

(15) Close CF access *n* *queueName*

The number of times a close caused a request to the CF to update the status in the Coupling Facility. This occurs when this is the last close on the queue manager, and the queue manager updates the CF to indicate that it does not have the queue open.

(15) Close No CF access *n* *queueName*

The number of closes which did not cause a CF update, because other applications on this queue manager have the queue open.

(15) Close topic srb CPU time *n* *queueName*

When a topic is closed it may attach an SRB to do some processing. This is the average SRB time per close request. It is calculated at the total SRB for close/count of close requests.

Getting from a queue

(5) *Get count* *n* *queueName*

The total number of get requests from the queue.

(5) *Get avg elapsed time* *n uS* *queueName*

The average elapsed time getting messages from the queue. This is calculated as the total elapsed time for gets/count of get requests.

(5) Get avg CPU time *n uS* *queueName*

The average CPU time getting messages from the queue. This is calculated as the total CPU time for gets/count of get requests.

(5) Get suspended time *n uS* *queueName*

The average time per message that the request was suspended within MQ. This includes logging of out of syncpoint gets, waits for internal latches etc.

(5) Get skipped message count *n* *queueName*

If a get request is made for a specific message, and the queue is not indexed, the queue is searched for the specified message. This field is the number of messages skipped over. This can also occur if another application is getting a message, so the message is unavailable. In this case the number of skipped messages per get should be small.

(10) Get pageset total count *n* *queueName*

The number of get requests which had to read from the page set.

(10) Get pageset elapsed time *n* *queueName*

If the get pageset count > 0, this is the average time spent getting the messages from the page set.

(15) Get log force count *n* *queueName*

Some gets can cause a log force, for example, if the buffer pool is critically short of buffers. This value is the number of times a log force occurred.

(15) Get log force elapsed time *n* *queueName*

If a log force occurred during a get, this is the average time per message doing a log force.

(17) Get log write count *n* *queueName*

This is the number of log write requests.

(17) Get log write elapsed time n queueName

If the application had to log data during the get request, this is the time taken for this write request. For requests within sync point, this just writes to log buffers.

(14) Get total empty pages n queueName

If all the messages on a page have been got, but the page has not been freed up, the page counts as empty. A large value can indicate an application getting a specific message from a deep queue, and the queue is not indexed.

(14) Get TOQ average n uS queueName

This is the average Time On the Queue. This is the time from when the message was put to the queue on this queue manager, to the time it was destructively got.

(14) Get TOQ maximum n uS queueName

This is the maximum time on the queue. This is the time from when the message was put to the queue on this queue manager, to the time it was destructively got.

(14) Get TOQ minimum n uS queueName

This is the minimum time on the queue. This is the time from when the message was put to the queue on this queue manager, to the time it was destructively got.

(14) Get valid count n queueName

This is the number of gets requests with return code zero, or truncated message accepted.

(10) Get size maximum n bytes queueName

This is the maximum size of message processed.

(10) Get size minimum n bytes queueName

This is the minimum size of message processed.

(10) Get size average n bytes queueName

This is the total number of bytes processed/count of valid gets (rc = 0 or MQRC_truncated_msg_accepted).

(10) Get Dest-Specific n queueName

The number of times a destructive get for a specific message id or correld-id was issued.

(10) Get Dest-Next n queueName

The number of times a destructive get for the first or next message was issued.

(10) Get Browse-Specific n queueName

The number of times a browse for a specific message id or correldid was issued

(10) Get QSUBrowse-Next n queueName

The number of times a browse for the first or next message was issued

(15) Get log force elapsed time n queueName

If a log force occurred during a get, this is the average time per message doing a log force.

(10) Get errors n queueName

This is for get requests which had reason codes MQRC_OPTIONS_ERROR,MQRC_GMO_ERROR, or MQRC_HOBJ_ERROR

(10) Get persistent count n queueName

The number of persistent messages processed

(10) Get non persistent count n queueName

The number of non persistent messages processed. This is calculated as count of valid

messages – count of persistent messages

Putting to a queue

(5) *Put count* *n* *queueName*
The number of puts to the queue.

(5) *Put avg elapsed time* *n uS* *queueName*
The average elapsed time putting messages to the queue. This is calculated as the total elapsed time for puts/count of put requests.

(5) *Put avg CPU time* *n uS* *queueName*
The average CPU time putting messages to the queue. This is calculated as the total CPU time for puts/count of put requests.

(15) *Put suspended time* *n uS* *queueName*
The average time per message that the request was suspended within MQ. This includes logging of out of syncpoint puts, waits for internal latches etc.

(15) Put pageset count *n* *queueName*
The number of put requests which put directly to the page set. If this value is non zero then the buffer pool is likely to have been constrained.

(15) Put pageset elapsed time *n* *queueName*
If the put pageset count > 0, this is the average time spent putting the messages to the page set.

(15) Put log force count *n* *queueName*
Some puts can cause a log force, for example, if the buffer pool is critically short of buffers. This value is the number of times a log force occurred.

(15) Put log write total count *n* *queueTime*
This is the number of log write requests.

(15) Put log write elapsed time *n* *queueName*
If the application had to log data during the put request, this is the time taken for this write request. For requests within syncpoint, this just writes to log buffers.

(15) Put + put1 valid count *n* *queueName*
This is the number of MQPUT +MQPUT1 requests which were successful, and had return codeMQCC_OK.

(15) Put waiting getter *n* *queueName*
This is the number of put requests which satisfied a waiting getter.

(15) Put topic srb CPU time *n* *queueName*
When putting to a topic, it may attach an SRB to do some processing. This is the average SRB time per put request. It is calculated at the total SRB for put/count of valid put requests.

(5) Put1 count *n* *queueName*
The number of puts to the queue.

(5) Put1 avg elapsed time *n uS* *queueName*
The average elapsed time putting messages to the queue. This is calculated as the total

elapsed time for puts/count of Put1 requests.

(5) Put1 avg CPU time n uS queueName
The average CPU time putting messages to the queue. This is calculated as the total CPU time for puts/count of Put1 requests.

(15) Put1 suspended time n uS queueName
The average time per message that the request was suspended within MQ. This includes logging of out of syncpoint gets, waits for internal latches etc.

(15) Put1 pageset count n queueName
The number of Put1 requests which Put1 directly to the page set. If this value is non zero then the buffer pool is likely to have been constrained.

(15) Put1 pageset elapsed time n queueName
If the Put1 pageset count > 0, this is the average time spent putting the messages to the page set.

(15) Put1 log force count n queueName
Some puts can cause a log force, for example, if the buffer pool is critically short of buffers. This value is the number of times a log force occurred.

(15) Put1 log force elapsed time n queueName
If a log force occurred during a Put1, this is the average time per message doing a log force.

(15) Put1 log write total count n queueTime
This is the number of log write requests.

(15) Put1 log write elapsed time n queueName
If the application had to log data during the Put1 request, this is the time taken for this write request. For requests within syncpoint, this just writes to log buffers.

(15) Put1 + put1 valid count n queueName
This is the number of MQPUT +MQPUT1 requests which were successful, and had return code MQCC_OK.

(15) Put1 waiting getter n queueName
This is the number of put1 requests which satisfied a waiting getter.

(15) Put1 topic srb CPU time n queueName
When putting to a topic, it may attach an SRB to do some processing. This is the average SRB time per Put1 request. It is calculated at the total SRB for put1/count of valid put1 requests.

(10) Put size maximum n queueName
This is the maximum message size from a put or PUT1 request.

(10) Put size minimum n queueName
This is the minimum message size from a put or PUT1 request.

(10) Put size average n queueName
This is the total number of bytes processed/count of valid puts or put1s (rc = 0).

(10) Put num persistent n queueName
This is the number of persistent messages put to the queue.

(10) Put num not peristent n queueName
This is the number of non persistent messages put to the queue. It is calculated as number of puts – number of persistent messages.

(10 Published msgs n queueName

This is the number of messages which resulted in a publish. A put or a put1 can result in 0 or more messages published.

MQINQ on a queue

(5) Inq count n queueName

The number of inquires to the queue.

(5) Inq avg elapsed time n uS queueName

The average elapsed time doing MQINQ to the queue. This is calculated as the total elapsed time for MQINQ/count of MQINQ requests.

(5) Inq avg CPU time n uS queueName

The average CPU time doing MQINQ to the queue. This is calculated as the total CPU time for MQINQ/count of MQINQ requests.

MQSET to a queue

(5) Set count n queueName

The number of MQSET requests to the queue

(5) Set avg elapsed time n uS queueName

The average elapsed time doing MQSET to the queue. This is calculated as the total elapsed time for MQSET/count of MQSET requests.

(5) Set avg CPU time n uS queueName

The average CPU time doing MQSET to the queue. This is calculated as the total CPU time for MQSET/count of MQSET requests.

(15) Set log force elapsed time n queueName

This is the average time per message doing a log force.

(15) Set log write elapsed time n queueName

This is the average time per message doing a log write.

Other information

(10) Curdepth maximum n queueName

This is the maximum depth of the queue found during the interval.

*(*) Total Q CPU used n us queueName*

This is the total of the CPU used to process this queue. It is the sum of the CPU used in the API requests.

*(*) Total Queue elapsed time n us queueName*

This is the total elapsed time processing this queue. It is the sum of the Elapsed time used in the API requests.

Additional information for shared queues

```
Open CF access                    2       APP1
Open No CF access                 0       APP1
Close CF access                   2       APP1
Close No CF access                0       APP1
```

Interpretation

```
(15) Open CF access                    2      APP1
(15) Open No CF access                 0      APP1
(15) Close CF access                   2      APP1
(15) Close No CF access                0      APP1
```

If the queue manager does not currently have a shared queue open, then when an application opens the queue, the queue manager has to update the CF to indicate it has the queue open. This is recorded as *Open CF access.* If there is an application with the queue open, then with another open request, the queue manager does not need to update the CF. This is recorded as *Open No CF access.* The Open CF access has a higher cost than the Open No CF access.

When the queue is closed, if there are other applications with the queue open, then there is no CF update. When the last application closes a queue, the queue manager updates the CF to indicate it does not have the queue open. There are two fields *Close CF No access* and *Close CF access*. A close with the *Close CF access* updates the CF and this has a slightly higher cost than a close with no CF access.

```
Get count                        1000       APP1
Get avg elapsed time             1096 uS    APP1
Get avg CPU time                   56 uS    APP1
CF time/verb    966
 CF Avg Sync elapsed time/verb      41 us
 CF Avg Sync number of request     140
 CF Avg Sync CF response time      294 us
 CF Avg Async elapsed time/verb    925 us
 CF Avg Async number of request    861
 CF Avg ASync CF response time    1074 us

  StartMon Avg Async elapsed time/verb      1 us
  StartMon Avg Async number of request      1
  StartMon Avg ASync CF response time    1253 us
  Move     Avg Sync elapsed time/verb      41 us
  Move     Avg Sync number of request     140
  Move     Avg Sync CF response time      294 us
  Move     Avg Async elapsed time/verb    924 us
  Move     Avg Async number of request    860
  Move     Avg ASync CF response time    1074 us
```

Interpretation

```
(10) CF time/verb    966
```

The `Get avg elapsed time` above was `1096 microseconds.` Of the average elapsed time of 1096 microseconds per get request, 966 microseconds were spent in a coupling facility request

```
(11) CF Avg Sync elapsed time/verb      41 us
(11) CF Avg Sync number of request     140
(11) CF Avg Sync CF response time      294 us
(11) CF Avg Async elapsed time/verb    925 us
(11) CF Avg Async number of request    861
```

```
(11) CF Avg ASync CF response time    1074 us
```

A coupling facility request can be synchronous or asynchronous. The times and counts is displayed for each type of request.
The total CF requests were 140 + 861 = 1001. Of these 140 requests were synchronous.
The average time spent doing synchronous requests was 294 microseconds.

Looking at the total number of get requests and allocating the synchronous and asynchronous time, on average each get request had 41 microseconds synchronous time, and 925 microseconds asynchronous. In reality a get was either synchronous or asynchronous – not a mixture. But this shows that most of the time of the get request 925 out of 1096 was spent in asynchronous requests.

```
(15)   StartMon Avg Async elapsed time/verb      1 us
(15)   StartMon Avg Async number of request      1
(15)   StartMon Avg ASync CF response time    1253 us
(15)   Move     Avg Sync elapsed time/verb      41 us
(15)   Move     Avg Sync number of request     140
(15)   Move     Avg Sync CF response time      294 us
(15)   Move     Avg Async elapsed time/verb    924 us
(15)   Move     Avg Async number of request    860
(15)   Move     Avg ASync CF response time    1074 us
```

This information is likely to be of use only to IBM personnel. The information displays the time at the detailed CF request level.
The information shows there was 1 StartMonitor request. This was an asynchronous request taking 1253 microseconds.
There were Move requests, some of which were synchronous and some were asynchronous.

Rules for accounting data

MQTASK01x Queue not indexed

Detail: 5
When: There are gets or browse from the queue where a message is got by msgid or correlid, but the queue is not indexed. If the queue is not indexed, this causes a sequential scan of the queue which can be expensive, the deeper the queue, the more expensive the request.
MQTASK01I is produced if the max queue depth is less than 10
MQTASK01W is produced if the max queue depth is greater equal than 10, and less than 100
MQTASK01E is produced if the max queue depth is greater equal than 100

Action: Review the queue definition, work with the applications team to determine how the queue should be indexed.

MQTASK02 High percent of no msg found

Detail: 12
When: The number of gets which did not return a message is more than 3 times the number of gets which returned a message.
When waiting for a particular message there will be a get for the message, and the message is not there, so the application waits. When the message arrives the application is posted and the get is reissued and retrieves the message-id.

Having a high proportion of requests not returning a message can indicate

1. A get next message is wanted, but the message-id or correlid field is not being cleared, and so the requests is a get for a specific message.
2. There are many application instances getting from the queue. A message arrives, all the applications rush to get the message. One instance is successful, the other instances are unsuccessful. This may not be a problem but may indicate you may need to look at the way the applications are set up.

Action: Review the applications.

MQTASK03 long open

Detail 10:
When: The average open time is greater than the Long_open parameter in the MQSMF formatting program.
Action: This may alert you to possible application problems. You should investigate the other data for the record, for example to see if there is a high latch wait – which would indicate contention with other applications, or with a long CF response time.

MQTASK04I long open ET >2 CT

Detail:10
When: Usually the elapsed time of an open request is close to the CPU used. This reports if the elapsed time is much longer than the CPU time used for the open.
Action: Investigate the delays.

MQTASK05I No msg returned

Detail: 8
When: There were gets from the queue, but none were successful. This is valid, but may indicate a problem with the application setup.
Action: Review the application.

MQTASK07I High browse rate/get ratio

When: There were gets and browses from the queue, and the number of browse requests is 3 times the number of get requests. This is valid, but may indicate a problem with the application setup.
Action: Review the application.

MQTASK08E Long Put time due to logging

Detail: 12
When: The average time for a put was longer than the Long_Put parameter in the MQSMF formatting program, and the average message size was less than 10,000 bytes. More than 20% of the time was spent logging.
Action: Review the logging statistics and the logging datasets to see if the system is constrained by the DASD.

MQTASK09E Long Put time not due to logging

Detail: 15
When: There average time for a put was longer that the Long_Put parameter in the MQSMF formatting program, and the average message size was less than 10,000 bytes. Less than 20% of the time was spent logging.

Action: You should examine the detailed record and find why the put took so long.

MQTASK10W Max depth > n and puts

Detail: 10
When: There maximum depth of the queue was greater than MaxDepth parameter in the MQSMF formatting program, and there were puts to the queue.
Action: Investigate why the queue depth is so large.

MQTASK12x Get Specific and Get Next

Detail: 5
When: There is a mixture of Get Specific and Get Next requests from an application. This is valid, but may not be as designed.

MQTASK12I - the max depth is less than 10 messages
MQTASK12I -the max depth is greater equal to 10 and less than 100
MQTASK12I -the max depth is greater equal 100
Action: Investigate the application.

MQTASK16S long latch wait n Name s

Detail: 15
When: A latch wait time was found where the latch time was greater than the 10* `LongLatchWait` parameter in the MQSMF formatting program.
Action: This is usually of interest only to IBM personnel when investigating performance problems

MQTASK16E long latch wait n Name s

Detail: 10
When: A latch wait time was found where the latch time was greater than the `LongLatchWait` parameter in the MQSMF formatting program.
Action: This is usually of interest only to IBM personell when investigating performance problems

TaskElapsed Time

This section has a summary of the tasks records, and where the time is spent.

This is written to ddname //TASKET.
The fields columns are

CPU	This is the sum of the CPU recorded.
LogLatch	This is the time waiting for the log latch
Latch	This is the time waiting for other latches
Log	This is the time waiting for log I/O activity – typically commits and out of syncpoint puts of persistent messages
PutPS	The time putting to a page set
GetPS	The time getting from a page set
Delta	The difference between the total time and the sum of the CPU, Log latch etc.. This could be application time, or time spent in MQ which is not recorded in the class 3 accounting because ACCTQ is off

TotalElapsedTime,
Duration

This allows you to quickly see where the time is spent for an application, for example logging or doing page set I/O

TaskCSV

This section has a summary of the tasks records, one per task/transaction name and CPU used, amount of data logged, put and got in MB. Data for multiple instances of a transaction or a job are summarised into one record.

This is written to ddname //TASKCSV.

The time is the time the SMF record was created, rounded down to the hour boundary. So if a transaction ran at 10:04:02.20 it would be recorded as 10:00:00

'Date','Time','Type','Tran1','Tran2','Count','CPU S','logBytesMB','put MB','Get MB',

2013/02/15,10:00:00,B,'PAICEP4',' ', 2,4.3,2096.2, 2000, 0,
2013/02/15,10:00:00,C,'CN15 ','IYFFC000',32091,2.3, 0, 20.1,30.2,
2013/02/15,10:00:00,C,'CP15 ','IYFFC000', 496,0.1,1.1697, 1.0, 0,
2013/02/15,10:00:00,M,'P1.TO.P2','MQP2<1414>', 10,0.9,23.57, 23.0, 0,

Interpretation

Date	
Time	
Type	B for Batch, RB for RRS batch, C for CICS, M for Mover, CS for command server, I for IMS
Tran	Batch job name
	CICS transaction
	Channel name
Tran2	CICS region name
	IP address
Count	Number of task records found for this row
CPU S	CPU used in seconds
logBytesMB	Total amount of data logged in MB
put MB	Total amount of data put, in MB
Get MB	Total amount of data get, in MB

Task Sumary

In ddname //TASKSUM is a summary of messages produced when looking at the task and queue records.
This has data like

```
Record#    Count    Value Message
```

```
   2202       25    98908 MQTASK13E long commit time C,'CP15','IYFFC000',
     38        1   106347 MQTASK13E long commit time B,'PAICEP7A',' ',
```

This has the following meaning.

- There was a message `MQTASK13E long commit time C,'CP15','IYFFC000',`
  - It was produced 25 times
  - The largest value (of the commit time) was 98908 microseconds. This was at record 2202 in the input file
- There was a message `MQTASK13E long commit time B,'PAICEP7A',' ',`
  - It was produced 1 time,
  - the largest (only) value was 106347 in record 38 of the input file.

To investigate these in more detail you can use StartRecord=2202, LastRecord=2202 and Detail(20). This will give all maximum level of detail for the one record.

– End of document –