IBM MQ for z/OS

Administration – Introduction to OEMPUT

IBM Washington Systems Center
Lyn Elkins – elkinsc@us.ibm.com
Mitch Johnson – mitchj@us.ibm.com

Oh Dear, Oh Dear

- BORING Alert

Agenda

- OEMPUT an overview
  - A bit of history and explanation
- Testing a z/OS Queue Manager
  - I’ve just installed MQ for z/OS
    - How do I know it is functioning?
  - I have just upgraded a z/OS queue manager
    - I want to make sure nothing is broken
  - I need to estimate the costs when an applications wants to change:
    - Non-persistent to Persistent messages
    - Message Size
    - Private queue to Shared queue
- Testing client connections

OEMPUT
A general MQ for z/OS Test tool

OEMPUT – a general test tool

- OEMPUT was originally developed to exercise the Message Broker on z/OS
  - It would help identify potential bottlenecks in MQ
  - It was based on a program used by the development lab for testing
  - It includes some simple performance metrics as output
    - Transactions per second
    - Approximate CPU use
  - It provides several message attributes that can be set to test various scenarios
    - Persistent or nonpersistent
    - Message size
    - Message type
  - Load and pseudo-production testing can be done
    - Option to allow running for a set amount of time
    - Option to put a message then wait for X hundredths of a second

OEMPUT – Basic Information

- OEMPUT is now part of SupportPac MP1B
  - https://www.ibm.com/support/pages/mp1b-ibm-mq-interpreting-accounting-and-statistics-data-and-other-utilities
- Source is not delivered
  - Load module only
- Installation is part of MP1B
  - Download
  - FTP files to z/OS
  - RECEIVE the load library and sample JCL library

OEMPUT – Running the program – an MQPUT1 sample

- The Sample JCL, tailored for some testing I was doing looks like this
- The parms are:
  - -m – The queue manager
    - Note: You can use a QSG name for the connection, but it does not return the QSG member
  - -put1 – Use an MQPUT1 command for each put, if not specified OEMPUT will open the queue and do puts in a loop, then close
  - -q – the queue messages will be put to this queue
  - -s – the length of the messages
  - -n – the number of messages to be put
  - -fileDD – the DD name of the file used for the message contents

```
//*
//   SET M=QML1
//   SET Q=ELKINSC.TEST.MQPUT1
//   SET R=ELKINSC.TEST.MQPUT1
//   SET L=1000
//   SET N=25
//PUT01A EXEC PGM=OEMPUT,REGION=0M,
// PARM=('-m&M -put1 -q&Q  -s&L -n&N -fileDD:MIN ')
//SYSIN DD *
/*
//STEPLIB  DD DISP=SHR,DSN=MQPERF.OEMPUT.LOAD
//         DD DISP=SHR,DSN=MQ91#.SCSQLOAD
//         DD DSN=MQ905.SCSQANLE,DISP=SHR
//         DD DSN=MQ905.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//MIN    DD DISP=SHR,DSN=MQPERF.ELKINSC.JCL(MSGS)
//COR    DD DISP=SHR,DSN=MQPERF.ELKINSC.JCL(COR01)
//SUMMARY  DD SYSOUT=*
```

OEMPUT – Running the program – an MQGET sample

- The Sample JCL, tailored for some testing I was doing looks like this

```
//  SET M=QML1
//  SET Q=ELKINSC.TEST.MQPUT1
//  SET R=ELKINSC.TEST.MQPUT1
//  SET L=1000
//  SET N=25
//PUT01A EXEC PGM=OEMPUT,REGION=0M,
// PARM=('-m&M -r&Q  -s&L -n&N  ')
//SYSIN DD *
/*
//STEPLIB  DD DISP=SHR,DSN=MQPERF.OEMPUT.LOAD
//         DD DISP=SHR,DSN=MQ905.SCSQLOAD
//         DD DSN=MQ905.SCSQANLE,DISP=SHR
//         DD DSN=MQ905.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//MIN    DD DISP=SHR,DSN=MQPERF.ELKINSC.JCL(MSGS)
//COR    DD DISP=SHR,DSN=MQPERF.ELKINSC.JCL(COR01)
//SUMMARY  DD SYSOUT=*
```

- The parms are:
  - -m – The queue manager
    - Note: You can use a QSG name for the connection, but it does not return the QSG member
  - -r – the queue messages will be retrieved from
  - -n – the number of messages to be retrieved

OEMPUT – Other frequently used parameters

- -l – Loop – controls the number of messages put and then gotten in a loop. Batches messages. The default value is 1, but if emulating an application it may put 10 messages, then start getting 10 ‘replies’
- -c – Commit – if the messages are being processed in loops (as above) will commit the ‘loop’
- -cgcpc – After any initial load of messages, this GETs the number of messages in the loop and commits, then puts the number of messages in the loop and commits
  - Designed to emulate client processing, though does not do the connects and disconnects
- -cgpc - Any initial load of messages, GET the number of messages in the loop, then puts the number of messages in the loop THEN commits.
  - Designed to emulate a long running server application
- -np or -p – Messages are Nonpersistent (the default) or Persistent
- -putwait – there is a wait of X hundredths of seconds between MQPUTs
- -tm or –ts – run the test for the specified number of minutes (-tm) or seconds (-ts)
- -w – the MQGET wait interval in seconds, default is 60 seconds

OEMPUT – Output from the PUT1 Test

```
Compiled Nov  1 2018 19:44:10.
parm: -mQSGM -put1 -qELKINSC.TEST.MQPUT1 -s1000 -n25 -fileDD:MIN
Message file -FILE: DD:MIN open mode:rb
bytes read from msg file  800
reply size 104857600
OEMPUT about to MQCONN to QMgr QSGM.
CPU type 0000013906
Date Time 2020/06/08 18:23:23.
Using MQPUT1
Entering PUT only loops...
Preload the queue with 0 messages...
  Message size        : 1000
  Reply size          : 104857600
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages      : 25
  Syncpoints          : NO-SYNCPOINT
Starting loop at 2020-06-08 18:23:23.280916
```

- The parameters used to control the job are listed.
- Note that even though there were no replies expected or included, a ‘reply size’ is listed
- It indicates that MQPUT1 is being used
- Then provides information about the test
  - Message persistence
  - Messages per loop
  - Total messages
  - Syncpoint status

OEMPUT – Output from the PUT1 Test - Continued

```
Total Transactions  : 25
Elapsed Time        :    0.001 seconds
Application CPU Time:    0.001 seconds  (85.1%)
Transaction Rate    : 20341.937 trans/sec
---------------------------------------------------

Round trip per msg  :       49 microseconds
Avg App CPU per msg :       41 microseconds
---------------------------------------------------
```

- Now for the interesting bits
  - The total transactions and the calculated transaction rate.
    - In this case there were only 25 messages put so the elapsed time and CPU time are not terribly interesting
  - The round trip per message and average CPU per message can be very useful when making comparisons – there will be more on that!

Testing a z/OS Queue Manager

Some ideas

Testing a new or upgraded z/OS queue manager

- At the WSC we use OEMPUT to test new queue managers
  - Simple tests to put and get
  - Nothing fancy
- There are samples delivered with MQ to test for additional functionality
  - Batch, CICS and IMS
- Running comparisons between an existing queue manager and an upgraded version can be helpful to see where there may be improvements
- Running comparisons, like the three that follow, can help quantify cost differences
  - Remember – each environment may have substantial physical differences that can mean even more differences in run time costs

Simple Comparison – Persistent vs Nonpersistent

```
  Message size         : 1000
  Reply size           : 10240
  Message persistence  : PERSISTENT
  Messages per loop    : 1
  Total messages       : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by     : Any message
Starting loop at 2020-06-08 21:41:54.066292
 Workload manager data
                  Samples %idle %unknown(MQ?) %using CPU %doing I/O %
   QML1CHIN.005E       35   100             0           0           0
   QML1MSTR.005D        9     0             0          88           0
             WLM queue delay 11
-------------------------------------------------------
Total Transactions   : 23318
Elapsed Time         :   10.000 seconds
Application CPU Time:    1.095 seconds  (11.0%)
Transaction Rate     : 2331.715 trans/sec
-------------------------------------------------------
Round trip per msg   :      428 microseconds
Avg App CPU per msg  :       46 microseconds
```

```
  Message size         : 1000
  Reply size           : 10240
  Message persistence  : NON-PERSISTENT
  Messages per loop    : 1
  Total messages       : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by     : Any message
Starting loop at 2020-06-08 21:40:06.311459
 Workload manager data
                  Samples %idle %unknown(MQ?) %using CPU %doing I/O %Wait for CPU
   QML1CHIN.005E       4   100             0           0           0             0
   QML1MSTR.005D       4    75             0          25           0             0
-------------------------------------------------------
Total Transactions   : 218063
Elapsed Time         :   10.000 seconds
Application CPU Time:    8.056 seconds  (80.6%)
Transaction Rate     : 21806.036 trans/sec
-------------------------------------------------------
Round trip per msg   :       45 microseconds
Avg App CPU per msg  :       36 microseconds
```

Simple Comparison – 10,000 bytes vs 1,000 bytes

```
  Message size        : 10000
  Reply size          : 10240
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages      : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by    : Any message
Starting loop at 2020-06-08 22:39:07.604818
 Workload manager data
                  Samples %idle %unknown(MQ?) %using CPU
   QML1CHIN.005E        8   100             0         0
   QML1MSTR.005D        8   100             0         0
-----------------------------------------------------------
Total Transactions  : 11241
Elapsed Time        :   10.001 seconds
Application CPU Time:    0.538 seconds  (5.4%)
Transaction Rate    : 1124.011 trans/sec
-----------------------------------------------------------
Round trip per msg  :      889 microseconds
Avg App CPU per msg :       47 microseconds
```

```
  Message size        : 1000
  Reply size          : 10240
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages      : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by    : Any message
Starting loop at 2020-06-08 21:40:06.311459
 Workload manager data
                  Samples %idle %unknown(MQ?) %using CPU %doing I/O %Wait for CPU
   QML1CHIN.005E        4   100             0         0          0             0
   QML1MSTR.005D        4    75             0        25          0             0
-----------------------------------------------------------
Total Transactions  : 218063
Elapsed Time        :   10.000 seconds
Application CPU Time:    8.056 seconds  (80.6%)
Transaction Rate    : 21806.036 trans/sec
-----------------------------------------------------------
Round trip per msg  :       45 microseconds
Avg App CPU per msg :       36 microseconds
```

Simple Comparison – Private vs Shared Queue

```
  Message size        : 1000
  Reply size          : 10240
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages      : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by    : Any message
Starting loop at 2020-06-08 21:40:06.311459
 Workload manager data
                Samples %idle %unknown(MQ?) %using CPU %doing I/O %Wait for CPU
   QML1CHIN.005E        4   100            0          0          0             0
   QML1MSTR.005D        4    75            0         25          0             0
-----------------------------------------------------
Total Transactions  : 218063
Elapsed Time        :   10.000 seconds
Application CPU Time:    8.056 seconds  (80.6%)
Transaction Rate    : 21806.036 trans/sec
-----------------------------------------------------
Round trip per msg  :       45 microseconds
Avg App CPU per msg :       36 microseconds
```

```
  Message size        : 1000
  Reply size          : 10240
  Message persistence : NON-PERSISTENT
  Messages per loop   : 1
  Total messages      : -1
  Syncpoints Get 1,  Put 1, Commit in syncpoint
  MQGET replies by    : Any message
Starting loop at 2020-06-08 22:31:02.630880
 Workload manager data
                Samples %idle %unknown(MQ?) %us
   QML1CHIN.005E       28   100            0
   QML1MSTR.005D       31    61            0
-----------------------------------------------------
Total Transactions  : 77578
Elapsed Time        :   10.000 seconds
Application CPU Time:    3.485 seconds  (34.8%)
Transaction Rate    : 7757.647 trans/sec
-----------------------------------------------------
Round trip per msg  :      128 microseconds
Avg App CPU per msg :       44 microseconds
```

Testing client connections

- Unfortunately OEMPUTX does not connect as a client will, but there are options
  - Using the sample amqsputc and amqsgetc client programs can demonstrate the costs of clients connecting directly.
  - A very simple example, and windows batch file:
    - Set the MQSERVER environment variable to set the SVRCONN variable to appropriate values
    - An execution of amqsputc to a test queue
    - An execution of amqsgetc to pull the messages from the test queue
    - It looks like this:

```
SET MQSERVER=ELKINSC.SVRCONN/TCP/9.82.31.252(1417)
amqsputc ELKINSC.TEST.MQPUT QML1 <D:\test10.txt >D:\putcout10.txt
amqsgetc ELKINSC.TEST.MQPUT QML1 >D:\getcout10.txt
```

Results from the test:

```
D:\Tests>SET MQSERVER=ELKINSC.SVRCONN/TCP/9.82.31.252(1417)

D:\Tests>amqsputc ELKINSC.TEST.MQPUT QML1  0<D:\test10.txt 1>D:\putcout10.txt

D:\Tests>amqsgetc ELKINSC.TEST.MQPUT QML1  1>D:\getcout10.txt
```

Input File

```
this is message 1
this is message 2
this is message 3
this is message 4
this is message 5
this is message 6
this is message 7
this is message 8
this is message 9
this is message 10
```

Output File

```
Sample AMQSGET0 start
message <this is message 1>
message <this is message 2>
message <this is message 3>
message <this is message 4>
message <this is message 5>
message <this is message 6>
message <this is message 7>
message <this is message 8>
message <this is message 9>
message <this is message 10>
no more messages
Sample AMQSGET0 end
```

In conclusion

- There is much, much more!
- But I hope this has given enough background to be useful.
  - Or enough to make a new admin familiar with some areas to explore.