Taking a performance view to running the MQ Console on z/OS

tonySharkey |Apr 27 2017 Updated

The IBM MQ Console is a browser-based MQ administration and monitoring tool that uses WebSphere Liberty server.

The MQ Console runs on an LPAR and can be used to administer all queue managers of a comparable level on the same LPAR.

Users of the MQ Console may configure a number of widgets to allow them to monitor and administer particular objects such as queues and channels for a specific queue manager.

The MQ Console connects to each queue manager in bindings mode, so no channel initiator needs to be running in order to use the MQ Console.

Tests run have used the MQ Console with large numbers of objects defined, such as 50000 queues and with relatively small numbers of concurrent users (50).

**Configuring the Queue Manager(s)**

**Queue Placement**

The MQ Console uses the "SYSTEM.REST.REPLY.QUEUE" to store the reply messages to its inquiries.

Whilst messages put to this queue do have an expiry time, in the event of large numbers of objects being queried or large numbers of concurrent inquiries, the queue depth can grow quite significantly.

As a result, it is advisable to consider the placement of this queue with regards to page set and buffer pool usage so as to ensure that deep queues do not impact business transactions.

**Enhanced Accounting and Statistics Trace**

When the MQ Console inquires against an MQ Queue Manager it is frequently connecting, performing a small amount of PCF work and then disconnecting. This model results in 1 SMF type 116 record being written per connection.

In an environment where information about 1000 channels is being retrieved, more than 2000 SMF type 116 records would be written for each browser refresh.

Widgets other than the channel widget do tend to collect their data in fewer tasks but there will still be multiple ‘other’-type calls for each connection.

Depending on the type of widgets configured, the number of objects being returned and the frequency of inquiry, there could be a significant increase in the number of task records written and it may be advisable to consider where class 3 accounting trace is appropriate for extended periods when the MQ Console is connected to the queue manager.

**Configuring the MQ Console**

There are a number of factors which may influence the cost and performance of the MQ Console on z/OS:

Liberty is a Java™ product and as such is largely eligible for offloading on systems which can **exploit available** **zIIP specialty** processors.

* How much offload achieved will depend on the number of specialty processors available, how busy they are and whether the LPAR has been configured to allow zIIP-eligible work to run on regular CP-type processors.
* The costs of satisfying the inquiries from the MQ Console in the MQ queue manager address space are not eligible for zIIP offload.
* Based on the RMF™ Workload Activity reports, between 60 and 95% of the MQ Console address space costs are eligible for zIIP offload.

As a Java™ product, **ensuring the heap is of sufficient size**is important to the performance. Guidance on how to alter the size of the Java heap for the WebSphere Liberty server can be found [here](http://www-01.ibm.com/support/docview.wss?uid=swg21596474).

* Using this advice, we created the jvm.options file in the &USERDIR./servers/mqweb directory.
* In our tests we set the heap to 4GB e.g. -Xmx4G and -Xms4G.

In order to utilize 64-bit storage for the heap, i.e. where more than 1500MB of heap was required, it is necessary to the MEMLIMIT attribute e.g.



When the MQ Console did exhaust its heap, the MQ Console job log reported:



* Note the location of the system dump relates to the owner of the started task rather than the more generic "DUMP" high level qualifier.
* In addition to the system dump, there were also a set of corresponding files in the &USERDIR./servers/mqweb directory.

Ensure the browser has **sufficient time to respond.**

* If the browser reports "command server time out", it is possible to increase the value by updating the mqwebusr.xml file to set a more appropriate value for the "readTimeout" variable e.g.

<variable name="readTimeout" value="120" />

Note that the units are seconds as per this [article](https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q128180_.htm).

**MQ Console Performance**

**Not all widgets are equal!**

* Our measurements suggest a cost per hundred objects to be of the order 7 CPU milliseconds (on z13) for most of the widgets, including queue, topic and subscription.
* The channel authentication widget is slightly more expensive at 27 CPU milliseconds per hundred but it is not envisaged that there will be so many channel authentication records as queues.
* The channel widget is the most expensive widget as it requires considerably more work to retrieve the required data. Costs observed were of the order of 0.5 CPU seconds per hundred channel objects returned.

**Browser-based filtering**

Each widget on the MQ Console allows a filter to be applied, which can be used to reduce the number of objects displayed in the window.

It should be noted that this filtering is applied at the browser, so that when a large number of the appropriate object type are defined, all objects will be returned, with the associated CPU cost of inquiry and network transport, before the browser applies the filter.

This may be a factor if the browser is connecting via a high-latency or low-bandwidth / congested network.

**How does the MQ Console scale with many MQ objects defined?**

The following example shows the increased cost of retrieving data as more objects are defined. As demonstrated in the following chart and table, the MQ Console costs are linear up to 30,000 queues.

The data in the chart is expanded in the following table, but it should be noted that the cost in the MQ queue manager address space increases disproportionately as the number of queues defined increases. This was due to the buffer pool and page set hosting the "SYSTEM.REST.REPLY.QUEUE" being insufficiently sized during our measurements which meant that the messages were put and gotten synchronously from page set and the page set having to expand.

**Cost per refresh (CPU milliseconds)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Queues** | **0** | **1000** | **2000** | **3000** | **10,000** | **20,000** | **30,000** |
| MQ Console | 14.55 | 98.55 | 169.38 | 243.47 | 866.23 | 1802.84 | 2549.15 |
| MQ Queue manager | 0.81 | 14.72 | 32.18 | 53.02 | 313.79 | 1052.11 | 2406.57 |
| **Total** | **15.36** | **113.26** | **201.56** | **296.49** | **1180.02** | **2854.95** | **4955.72** |

**Note:** Costs shown are the total for the address spaces. If zIIP specialty processors were available, the MQ Console costs could be reduced by up to 95% of the quoted values.

**Increase in cost per refresh for each 1000 queues defined (CPU milliseconds)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Queues** | **0** | **1000** | **2000** | **3000** | **10,000** | **20,000** | **30,000** |
| MQ Console |   | 84 | 70.8 | 74.1 | 89 | 93.7 | 74.6 |
| MQ Queue manager |   | 13.91 | 17.5 | 20.8 | 26.1 | 73.8 | 135.5 |

The MQ console costs remain fairly consistent for each 1000 queues defined, although there is some variation which in part is due to the LPAR being lightly but variably loaded.

As previously mentioned the queue manager cost increases disproportionately with 20,000+ objects due to incorrect buffer pool and page set sizing.

**Round-trip times with increasing queue objects (seconds)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Queues** | **0** | **1000** | **2000** | **3000** | **10,000** | **20,000** | **30,000** |
| Time in seconds | 0.49 | 0.68 | 1.41 | 1.41 | 1.49 | 4.83 | 8.68 |
| Time in seconds with correctly sized buffer pool and page set |   |   |   |   | 1.49 | 3.43 | 5.45 |

**Note**: When the buffer pool and page set were made sufficiently large, the response time with 30,000 queues dropped by 38%.

This did not have an impact to the overall cost as there was a reduction in queue manager SRB of similar amount to the increase in queue manager TCB.

Tests were run with up to 70,000 queues. The limiting factor on our test systems was that with other objects defined, there were in total over 100,000 objects so the amount of virtual storage used to host those objects meant there was insufficient 31-bit private storage available to satisfy the MQ Console request.

**Conclusion**

The MQ Console requires some tuning, but is able to provide good response times for tens of concurrent users and at reasonable CPU costs, when used to administer queue managers with tens of thousands of objects.