IBM MQ - RDQM HA Performance Report

Version 1.0 - August 2018

Paul Harris
IBM MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire

Please take Note!

Before using this report, please be sure to read the paragraphs on "disclaimers", "warranty and liability exclusion", "errors and omissions", and the other general information paragraphs in the "Notices" section below.

First Edition, August 2018.

This edition applies to *IBM MQ V9.0.4* (and to all subsequent releases and modifications until otherwise indicated in new editions).

Note to U.S. Government Users

Documentation related to restricted rights.

Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

DISCLAIMERS

The performance data contained in this report was measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends on the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS

The information set forth in this report could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

INTENDED AUDIENCE

This report is intended for architects, systems programmers, analysts and programmers wanting to understand the performance characteristics of. IBM MQ RDQM capabilities, in comparison to MIQM. The information is not intended as the specification of any programming interface that is provided by IBM. It is assumed that the reader is familiar with the concepts and operation of IBM MQ, and RDQM.

LOCAL AVAILABILITY

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. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES

Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS AND SERVICE MARKS

The following terms used in this publication are trademarks of their respective companies in the United States, other countries or both:

- **IBM Corporation :** IBM
- **Oracle Corporation :** Java

Other company, product, and service names may be trademarks or service marks of others.

EXPORT REGULATIONS

You agree to comply with all applicable export and import laws and regulations.

Contents

1 Report Highlights

This report contains data points that illustrate the performance of the RDQM high availability (HA) solution delivered in the V9.0.4 CD, and V9.1 LTS releases of MQ for Linux. It is worth noting the following highlights:

- Over 43,000 round trips/second peak messaging rate in RDQM HA enabled scenario (~86,000 messages produced and ~86,000 messages consumed). See section 3.2.
- Peak messaging rate for RDQM HA is faster than MIQM equivalent. See section 3.
- RDQM HA scenarios with SSD backed DRBD storage, run up to 82-90% of rate achieved by a stand-alone QM, logging to SAN. See section 3.
- RDQM recovery from fail-over is faster for RDQM than MIQM in all scenarios tested (see section 5).

2 RDQM Introduction

RDQM based HA can be enabled by configuring three servers in an HA group, where each server has an instance of the queue manager, and the queue manager's data held in a DRBD backed filesystem. This file system was hosted on local SSDs for the purposes of the paper, giving optimum performance. Queue manager data is replicated across the servers by DRBD. An important distinction between the RDQM HA approach and MIQM is that there is not a single point of failure with regards to the queue manager data.

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q130280_.htm

If separate networks and switches (if required) are used to connect the three servers, then they can also continue to operate in the event of a partial network outage.

To ensure clients reconnect to the newly active QM on another server, the clients should be made aware of the IP addresses assigned to the workload interfaces of all three queue managers; or a Virtualised IP address in the case that a suitable load balancer component is employed.

RDQM HA has a "floating IP" feature that means that a client can be configured with a single IP address for an RDQM without the need for a load balancer or similar, if RDQM is employed in an environment where that could be used.

3 RDQM Throughput Tests

All the scenarios featured in this section utilise Requester/Responder messaging scenarios and the published messaging rate is measured in Round Trips/sec, which involves 2 message puts and 2 message gets. If you are only utilising one-way messaging (using a message sender, queue and message receiver to perform 1 message put and 1 message get), and you can avoid queue-lock contention, then you may achieve up to double the published rates.

The Requester/Responder test is detailed in : Workloads, and is presented here with results from running tests against three deployments of MQ:

- One or ten stand-alone queue manager(s), logging to SAN.
- One or ten MIQM queue manager(s) (with the NFS filesystem deployed on enterprise class SSDs on a server connected to the primary/standby, via 10Gb links).
- One or ten RDQM queue manager(s), where the three RDQM nodes are connected via 10Gb links and the filesystems are all deployed on enterprise class SSDs.

Appendix A details the machine configurations and specifications, used in these tests.

The version of the MQ tested in this section is V9.0.4.

Each test was conducted using a 2K (2048 byte) message size and this data is shown in the graphs included below. Additional tests were conducted using 20K and 200K to provide further data points.

3.1 Throughput Results for a Single Queue Manager

Results are presented for various numbers of requester application threads distributed across the 10 pairs of queues, 300 fixed responder application threads (30 responders per request queue) will send the replies to the appropriate reply queue, and the report will show the message rates achieved (in round trips/second) as the number of requesters is increased.

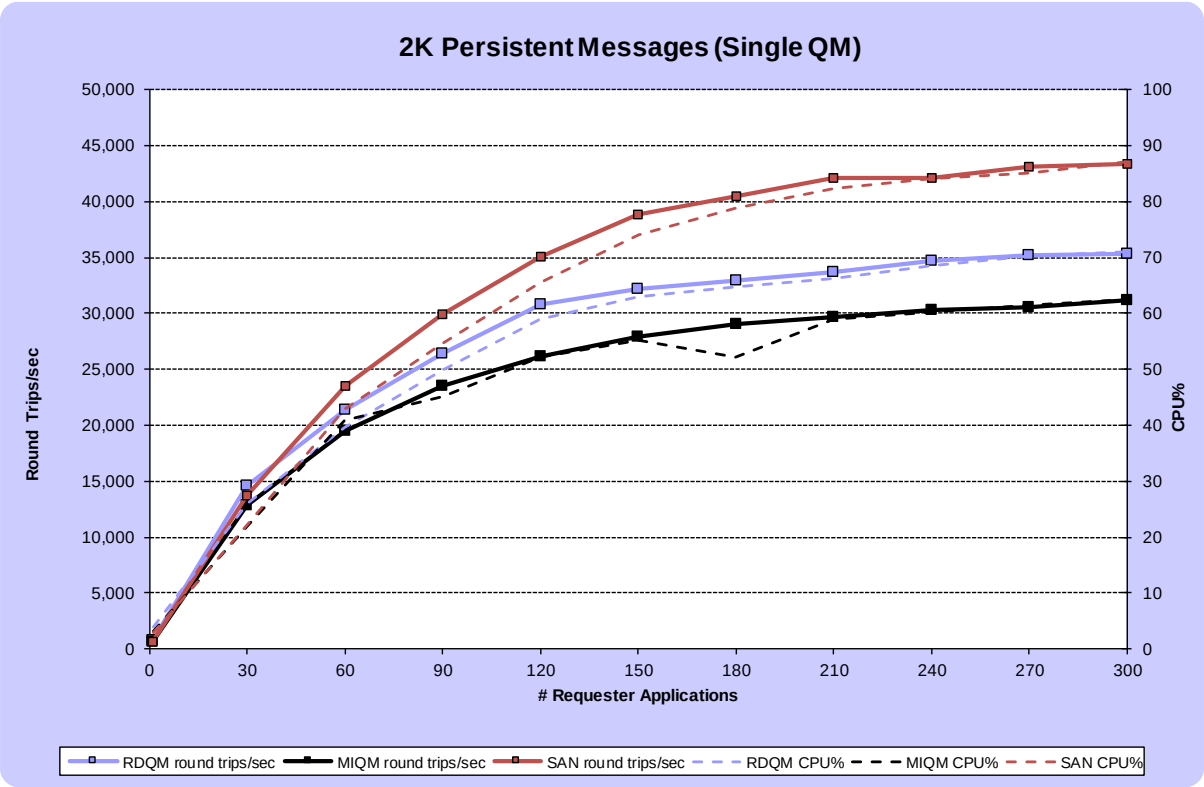

FIGURE 1 – PERFORMANCE RESULTS FOR 2KB PERSISTENT MESSAGING

Figure 1 shows that by enabling RDQM HA capability, the maximum throughput achieved with a 2K message size is reduced by approximately 18%, compared to a standalone QM, logging to SAN. There is a similar reduction in CPU utilisation. There is a greater disparity for larger messages (where the log writes become larger, offsetting the higher latency of the SAN filesystem), but RDQM out-performed the equivalent MIQM test in all cases.

Test	RDQM			MIQM			SAN		
	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients
2K Persistent Messages (Single QM)	35,379	71.13	300	31,134	62.38	300	43,351	87.05	300
20K Persistent Messages (Single QM)	10,756	28.64	300	7,102	20.38	240	19,110	45.56	300
200K Persistent Messages (Single QM)	1,140	9.58	30	815	6.7	30	1,937	11.4	30

*Round trips/sec

TABLE 1 - PEAK RATES FOR PERSISTENT MESSAGING

3.2 Throughput Results for 10 Queue Managers

This test repeats the one run in section 3.1, but spreads the load across 10 queue managers.

Results are presented for various numbers of requester threads distributed across the 10 Queue Managers who each host 10 pairs of queues (representing 10 applications per QM), 300 fixed responder threads (3 responders per request queue) will send the replies to the appropriate reply queue which are subsequently received by the originating requester threads, and the report will show the message rates achieved (in round trips/second) as the number of requesters is increased.

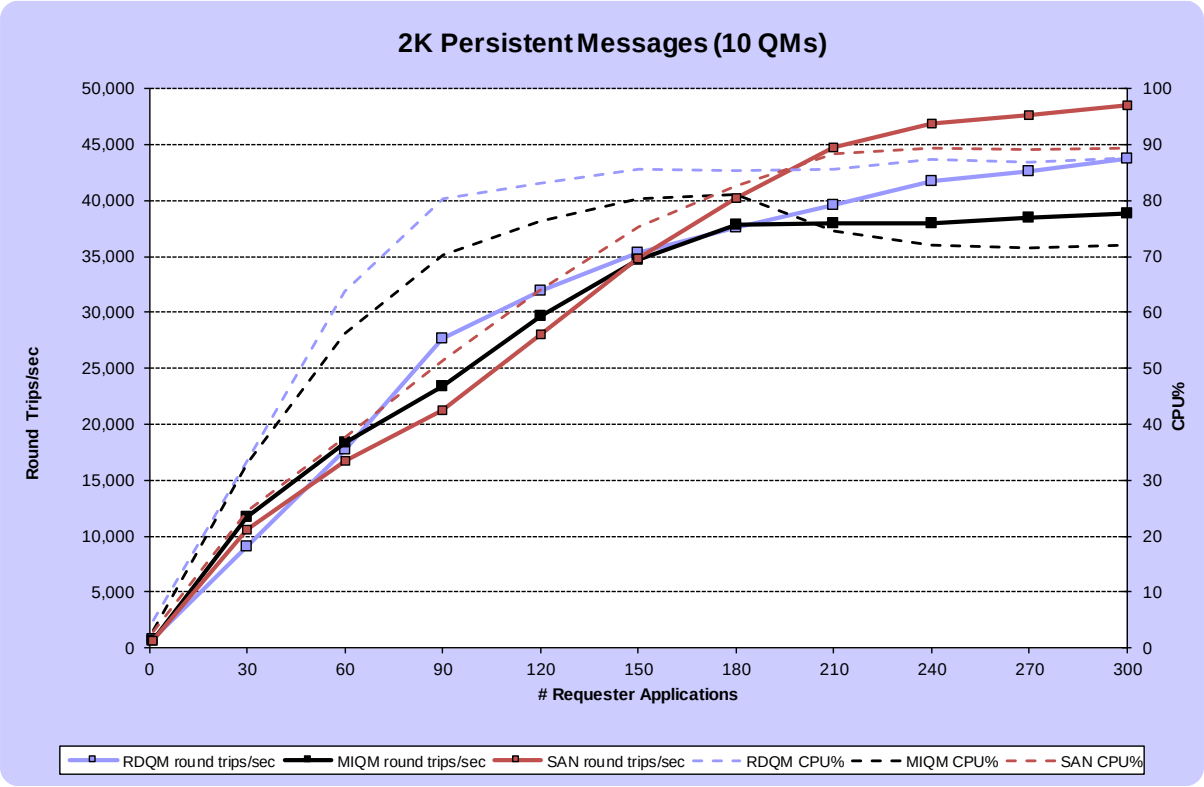

FIGURE 2 - PERFORMANCE RESULTS FOR 2KB, 10QM PERSISTENT MESSAGING

Figure 2 shows that when we have multiple QMs performing 2KB persistent messaging, the messaging rate is approximately 10% less than when distributed across a set of non-HA Queue Managers. Once again, there is a greater disparity for larger messages (where the log writes become larger, offsetting the higher latency of the SAN filesystem), but RDQM out-performed the equivalent MIQM test in all cases.

Test (Name : Type)	RDQM			MIQM			SAN		
	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients
2K Persistent Messages (10 QMs)	43,721	87.57	300	38,879	72.1	300	48,491	89.42	300
20K Persistent Messages (10 QMs)	13,370	55.28	270	11,618	57.31	90	22,719	59.12	300
200K Persistent Messages (10 QMs)	1,377	30.31	30	887	15.81	9	2,253	28.87	30

*Round trips/sec

TABLE 2 - PEAK RATES FOR 10QM PERSISTENT MESSAGING

4 How does HA perform over larger distances?

The previous section shows how the MQ HA capability, utilising RDQM might perform if all three of the RDQM nodes were in the same data centre (in our case, connected to the same local 10Gb switch for data replication). How would the HA performance differ if the nodes were located a larger distance apart? Due to testing limitations, we need to simulate the additional network delay that might be experienced as the distances between the nodes grows.

If the RDQM nodes are located 100Km apart, you might expect the smallest increase in packet transmission latency for each leg to be calculated as follows:

distance / speed = time

100,000m / 300,000,000m/s[a] = 0.000333s = 333 microseconds

There must also be an allowance for the refraction index of the cable

333 * 1.5 = 500 microseconds

Switching hardware and non-linear cable routing will likely further increase the latency between the nodes. The current advice to customers is that the network latency of the data replication links between RDQM nodes should be no greater than 5ms (though your own tolerance may be lower than this).

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q130980_.htm

A delay can be inserted into the sending network layer of each node's data replication link, to simulate such latency, allowing us to examine how this impacts the RDQM performance. The following chart repeats the test in section 3.2, and shows the effect of an additional 2ms round trip delay introduced into the network layer between the three RDQM nodes.

[a] Assuming speed of light to be 3x10⁸m/s

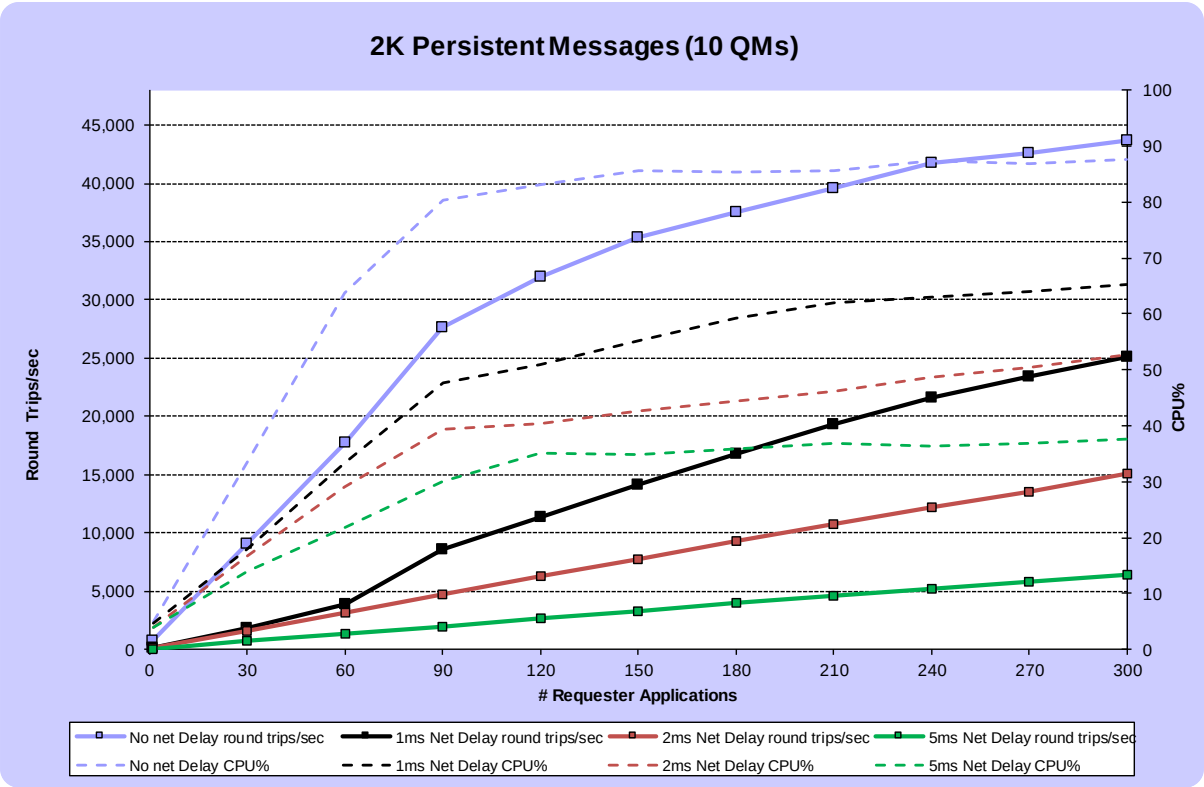

FIGURE 3 – RDQM PERFORMANCE RESULTS FOR 2KB, 10QM PERSISTENT MESSAGING WITH 0, 1MS, 2MS, & 5MS BASE NETWORK LATENCY (ROUND TRIP), ON DATA REPLICATION LINKS.

Figure 3 shows that an additional 1ms delay on the round-trip time of the HA data replication link results in a significant reduction in performance, when compared with the direct connection (no additional latency) between the RDQM nodes. As the delay increases, the impact becomes greater, as expected (a base 2ms delay on the data replication links reduces peak throughput from 43,721 round trips/sec to 15,041, for instance). Note that MQ will attempt to aggregate log data into larger disk writes, as the latency of the underlying file system increases, but this is dependent on higher levels of concurrency (more applications).

The data in the following tables show the full results for tests with an additional network delay of 1ms, 2ms & 5ms, across the range of message sizes tested.

The latency number in the charts below shows the average response time of each round trip (i.e. the time between sending a request and receiving a reply. This latency is dependent on the network delay, and the time it takes to write the log records for the transaction (which will increase with the size of the message).

Test	No net Delay			1ms Net Delay				
	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients	Latency #	Delta †
2K Persistent Messages (10 QMs)	43,721	87.57	300	25,077	65.22	300	6.3	57%
20K Persistent Messages (10 QMs)	13,370	55.28	270	11,724	50.65	270	6.7	88%
200K Persistent Messages (10 QMs)	1,377	30.31	30	1,279	31.88	27	8.9	93%

***Round trips/sec**
 # Single thread round-trip latency
† Percentage of 'no delay' rate.

TABLE 3 - PEAK RATES FOR 10QM PERSISTENT MESSAGING WITH ADDITIONAL 1MS SIMULATED NETWORK DELAY

Test	No net Delay			2ms Net Delay				
	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients	Latency	Delta †
2K Persistent Messages (10 QMs)	43,721	87.57	300	15,041	52.66	300	11.2	34%
20K Persistent Messages (10 QMs)	13,370	55.28	270	9,986	46.13	300	11.9	75%
200K Persistent Messages (10 QMs)	1,377	30.31	30	1,012	29.82	30	14	73%

*Round trips/sec
Single thread round-trip latency
† Percentage of 'no delay' rate.

TABLE 4 - PEAK RATES FOR 10QM PERSISTENT MESSAGING WITH ADDITIONAL 2MS SIMULATED NETWORK DELAY

Test	No net Delay			5ms Net Delay				
	Max Rate*	CPU%	Clients	Max Rate*	CPU%	Clients	Latency	Delta †
2K Persistent Messages (10 QMs)	43,721	87.57	300	6,436	37.52	300	25.6	15%
20K Persistent Messages (10 QMs)	13,370	55.28	270	4,932	35.04	300	27.2	37%
200K Persistent Messages (10 QMs)	1,377	30.31	30	512	28.51	30	40	37%

*Round trips/sec
Single thread round-trip latency
† Percentage of 'no delay' rate.

TABLE 5 – PEAK RATES FOR 10QM PERSISTENT MESSAGING WITH ADDITIONAL 5MS SIMULATED NETWORK DELAY

5 Fail-over Tests

How long does it take before the secondary QM node in an HA deployment becomes available, and how long will it be before the application(s) recover to their previous stable state?

Some scenarios were run to compare a MIQM with RDQM for idle QMs, versus busy QMs, and for QMs hosting deep queues.

First, some terminology:

The following section considers and compares fail-over times in an HA environment configured with either RDQM, or MIQM. We will use the following terms in either case.

Primary QM: The primary queue manager. This is the primary RDQM queue manager, or the 'active' MIQM queue manager.

Secondary QM: A secondary queue manager. This is a secondary RDQM queue manager, or the 'standby' MIQM queue manager.

When we talk about applications re-connecting to a secondary QM, we really mean the newly promoted QM (i.e. a secondary RDQM QM, that has now become the primary QM, or a standby QM in MIQM that has now become the active QM).

5.1.1 Fail-over Test Scenarios

Our JMS test application was configured to attempt to reconnect to the secondary QM, when any errors were received whilst communicating with the primary QM. How quickly the application detects an error depends on how the fail-over is triggered.

The test scenario is described in section 0, whilst : Topologies and Machine SpecificationsAppendix A describes the HA topologies, and machine specifications.

In our testing we triggered the fail-over in two ways, detailed in the following sections.

- Controlled 'Switch-over'

  The primary queue manager was switched to the secondary, by executing the appropriate command:

  For MIQM, executing the **endmqm -s** command will switch the queue manager over to the stand-by

  For RDQM, executing the **rdqmadm -p -m <QM>** command on a node hosting the secondary QM in the RDQM HA group, will switch the primary queue manager over to that node.

  Connected applications, will get an immediate error response to any outstanding MQ API calls (e.g. JMSWMQ2007 for 'send'). This test case was measured for:

  1. An idle queue manager (simplest case)
  2. Persistent message workload running at a total, target fixed rate of 50,000 PUT/GET pairs/sec against 10 'empty' queues (i.e. there are no messages

on the queues apart from those being put and got off the queue during the execution of this workload (busy case).

Note that each RDQM node only hosted a single HA queue manager. In other deployments an RDQM node might host, a primary HA QM for one RDQM HA group, and one or more secondary/tertiary QMs for other HA groups.

- Network failure

  A loss of the machine hosting the primary node/QM was simulated, by dropping all inbound/outbound packets for the links to the applications, the RDQM data replications link (or NFS link, in the case of MIQM), and the pacemaker link, effectively stopping all communications to the host. Iptables rules were used to control this.

  RDQM will fail-over the QM, when the broken data replication link, and loss of quorum is identified, whilst the JMS application may now be subject to longer delays, waiting on outstanding responses from MQ, which will not arrive (*our* application will now be dependent on heart-beating, controlled by HBINT to react to the loss of the primary QM). HBINT will trigger errors such as JMSWMQ2007 (MQPUT), JMSWMQ2002 (MQGET) or JMSCMQ0002 (MQCMIT), with a linked exception of:

  JMSCMQ0001: IBM MQ call failed with compcode '2' ('MQCC_FAILED') reason '2009' ('MQRC_CONNECTION_BROKEN').

  This test case was measured for:

  1. Persistent message workload running at a total, target fixed rate of 50,000 PUT/GET pairs/sec against 10 'empty' queues (i.e. there are no messages on the queues apart from those being put and got off the queue during the execution of this workload (same busy case as for the manual switch-over).
  2. Persistent message workload running at a total, target fixed rate of 50,000 PUT/GET pairs/sec against 10 deep queues (500,000 x2K messages on each queue). Deep queues, trigger queue loads on the queue manager switched to, which can have a significant impact on the start-up time.

5.1.2 Example Fail-over Timeline

There are several phases to consider when evaluating the time, it takes for an application to failover to a secondary queue manager:

i. Detection by RDQM, or MIQM that the primary QM is no longer responding.
ii. Making a secondary QM available, this involves replaying the transaction log to bring the queue files up to date (this will depend on the current persistent messaging rate and the time since last log checkpoint), and some queue loading.
iii. Time for client to notice disconnection (iiia) and reconnect to the secondary QM (iiib).
iv. Time to recover the prior message rate. Once the applications are re-connected, some additional queue loading may be triggered, if there were deep queues which were not involved in transactions recovery.

As an example, if a network failure occurred on a primary QM which was busy, and hosted deep queues, including live transactions, the timeline shown in Figure 4 might be typical. The green parts of the timeline are when the application is connected and running at its peak, steady state. The red parts indicate the time during which the messaging rate is 0 (QM is unavailable, or the application is waiting on HBINT/re-connection. The yellow part indicates a phase where applications are connected, but the

peak rate has not been achieved due to additional queue loading being caused by the application accesses.

It can be seen that, different phases of the failover may be taking place concurrently (such as the RDQM heartbeat timeout, and the HBINT, heartbeat timeout. In the example shown, HBINT causes the application to attempt connecting to the secondary QM before it is ready. If there are no deep queues then the secondary QM may well be open for business before the client has recognised that the connection to the primary QM has been lost, i.e. phase (ii) may occur before phase (iii). In this case, HBINT becomes the main factor in determining the overall time it takes from network failure to recovery of the peak, steady state.

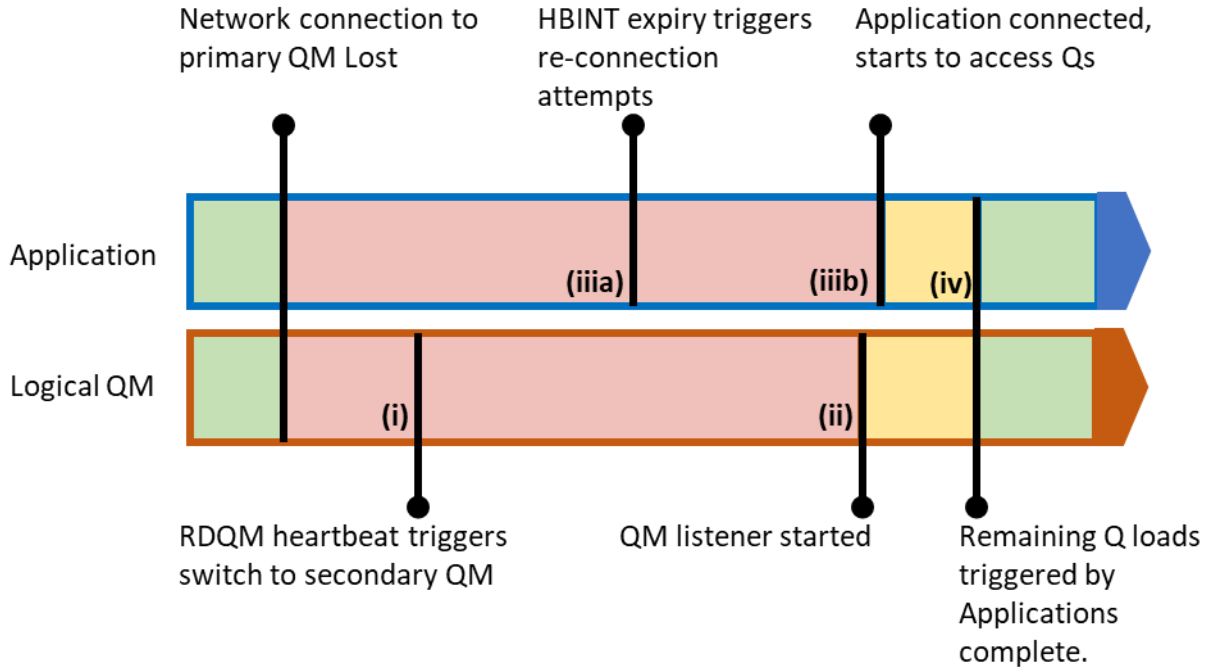

FIGURE 4 - EXAMPLE FAIL-OVER TIMELINE

5.1.3 Fail-over Results

For each of the tests detailed in the sections above, a QM re-start time was calculated from the MQ errors logs (basically the time from when the QM switch command, or Iptables command was issued, to the time the queue manager was connectable to again).

For 'busy' scenarios, where a workload was running, an application re-start time was calculated as the time taken from the client rate dropping to 0 to the rate recovering to pre-switch levels.

Results for all tests can be seen in Table 6, below.

Switch-over tests are those triggered by the **rdqmadm -p**, or **endmqm -s** commands.

Fail-over tests are those where packets on the network link are dropped. Note that in the fail-over test, the initial part of the QM re-start time is dependent on the HA mechanism (RDQM, or MIQM) detecting that the primary QM is no longer available. For RDQM, this is controlled by the underlying clustering technology (Pacemaker/Corosync), and results in a 10 second time-out. For MIQM, the time-out is dictated by the loss of the NFS lease on MQ files. For the tests run in this report, the default NFS lease (on the NFS server) was changed from 90 seconds to 10 seconds, to be more comparable to RDQM, as follows:

```
echo 10 > /proc/fs/nfsd/nfsv4gracetime
echo 10 > /proc/fs/nfsd/nfsv4leasetime
```

	RDQM			MIQM				
		Re-start times (mm:ss)			Re-start times (mm:ss)			
	Msg Rate	QM	Application	Msg Rate	QM	Application	Q Depth	HBINT
Idle Switch-over	n/a	00:02	n/a	n/a	00:03	n/a	0	n/a
Busy Switch-over	~50K/sec	00:04	00:14	~50K/sec	00:13	00:16	0	300
Busy Fail-over	~50k/sec	00:11	06:03	~50K/sec	02:36	06:05	0	300
Busy Fail-over (low HBINT)	~50K/sec	00:10	00:45	~50K/sec	03:06	03:09	0	20
Busy Fail-over (deep queues)	~50K/sec	01:21	01:23	~12K/sec	01:39	02:08	500,000	20

TABLE 6 - RE-START TIMES FOR QM AND APPLICATIONS

RDQM out-performs MIQM in all tests, largely because transaction recovery and queue loading are quicker from a local filesystem, than across the NFS link. The following patterns can be observed in the results:

Idle Switch-over — The secondary QM was available within 3 seconds of the switch being issued for both RDQM and MIQM.

Busy Switch-over — Application recovery was similar for RDQM & MIQM (14 & 16 seconds respectively).

Busy Fail-over — RDQM QM availability was established a lot faster (11 seconds, compared to 2minutes 36 seconds). With the default setting of 300 for HBINT however, the application recovery times were very similar, being dominated by the time taken to detect the network outage.

Busy Fail-over (low HBINT) — Reducing the value of HBINT to 20 enabled the applications to respond faster to the network fault (40 seconds), so the RDQM recovery time now benefits much more from the faster QM fail-over.

Busy Fail-over (deep queues) — With a lot of data on the queues, associated with outstanding transactions that occur when a sudden failure occurs (like the network outage simulated here), there is a lot more work to do to bring the system back to the same steady state. Much of this involves log replay and loading data from the queue files, which favours RDQM, with its local, fast storage. Note that the MIQM recovery time is less here than, with the previous test, but there will be less transactions to recover, due to the target rate of 50K/second PUT/GETs being unachievable, using such deeply populated queues.

Appendix A : Topologies and Machine Specifications

The same 5 machines were used for both the RDQM and MIQM testing.

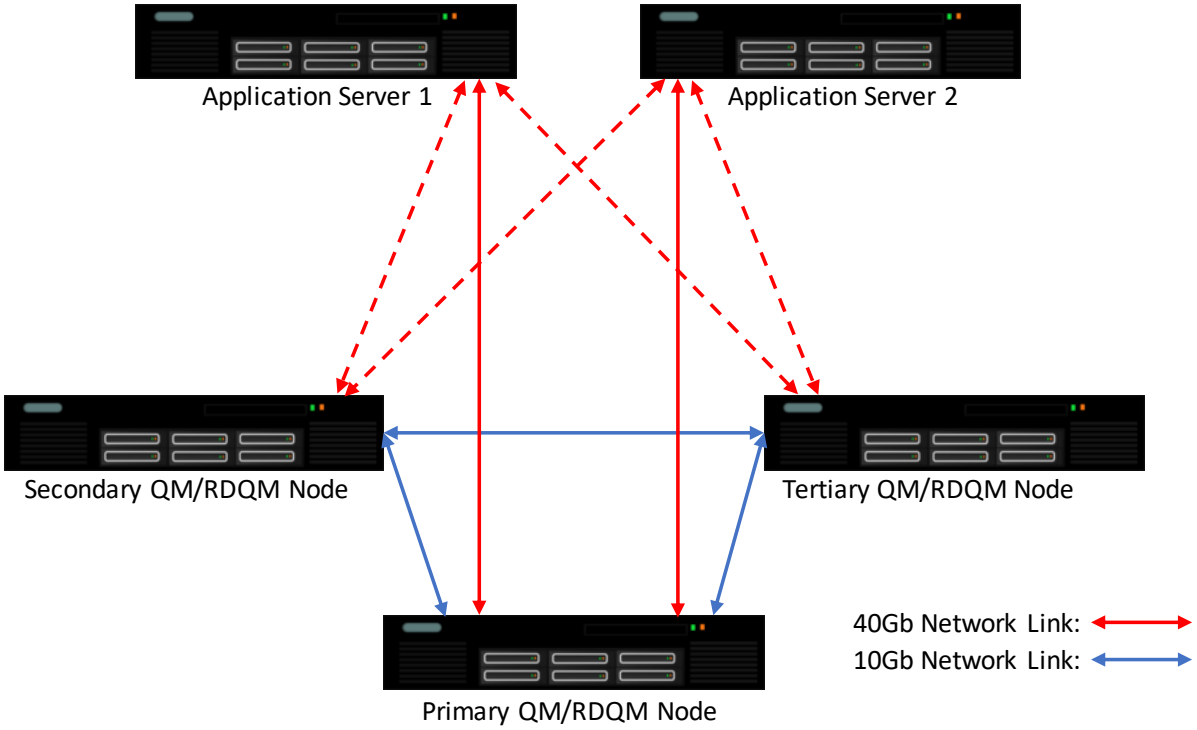

FIGURE 5 - RDQM TEST TOPOLOGY

For RDQM testing, all three RDQM nodes were of type 1 (see machine types, below), and the two application hosts were of type 2. The DRBD volume groups were deployed on a RAID0, enterprise SSD volume. Links between the applications and the RDQM nodes were 40Gb, whilst the RDQM data replications links were 10Gb. Each RDQM node had a single Pacemaker address (HA_Primary), utilising the 40Gb link. For failover testing, only one application host was used.

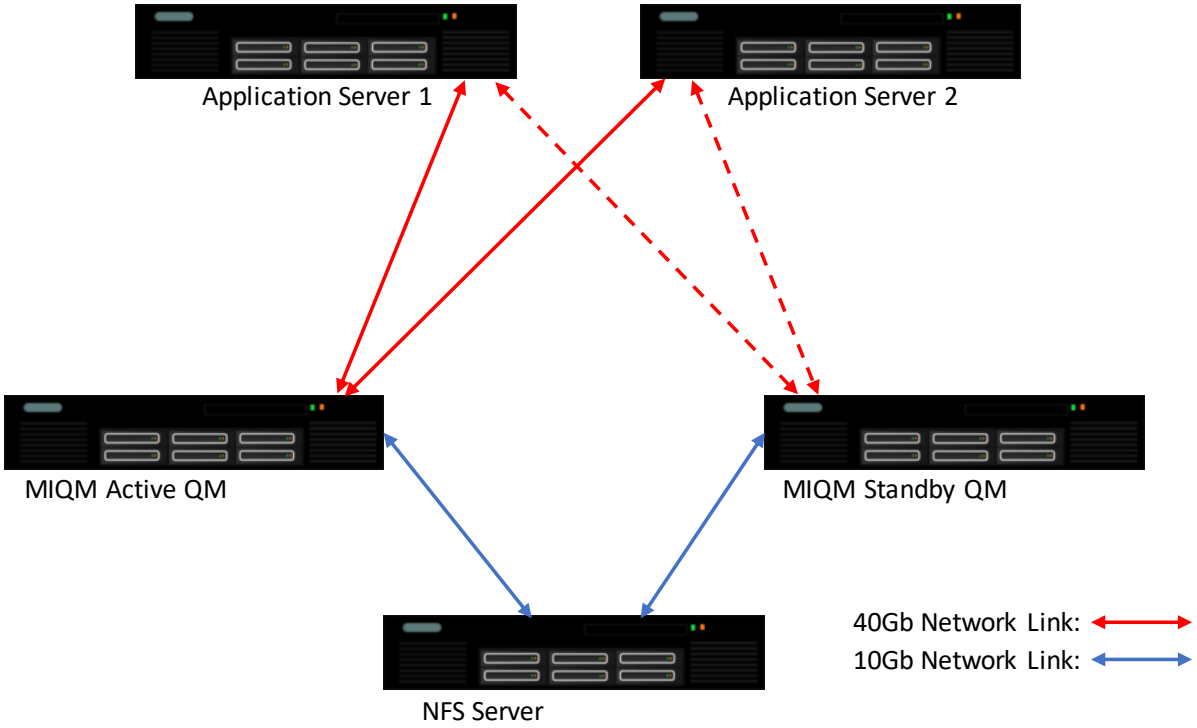

FIGURE 6 - MIQM TEST TOPOLOGY

For MIQM testing, the two MIQM QM hosts, and the NFS server were of type 1 (see machine types, below), and the two application hosts were of type 2. The file system exported by the NFS server to host the MQ logs and queues, was deployed on a RAID0, enterprise SSD volume. Links between the applications and the MIQM QM hosts were 40Gb, whilst the NFS links were 10Gb. For failover testing, only one application host was used.

Machine Types

Type 1 (RDQM nodes, MIQM Active/Standby hosts & NFS Server)

Category	Value
Machine	x3550 M5
OS	Red Hat Enterprise Linux Server 7.4
CPU	2 x 14 Cores: Intel Xeon E5-2690 V4 @ 2.6GHz.
RAM	128GB RAM
Network	1Gb, 10Gb & 40Gb Ethernet
Disks (hosting MQ logs/queues for RDQM nodes or as NFS server)	2 x 400GB, 6Gb SATA, Enterprise Performance SSDs (00YC326) in RAID 0 array.
RAID	ServeRAID M5210 (4GB Flash RAID cache)

Type 2 (Application Clients)

Category	Value
Machine	x3550 M5
OS	Red Hat Enterprise Linux Server 7.4
CPU	2 x 12 Cores: Intel Xeon E5-2690 V3 @ 2.6GHz.
RAM	128GB RAM
Network	1Gb, 10Gb & 40Gb Ethernet

Appendix B : Workloads

Two scenarios were used to collect the measurements in this report:

Test Scenario 1 – Requester/Responder (Persistent messages)

The test scenario in Figure 7 is a Requester/Responder scenario that simulates several applications that interact with a single QM. Ten pairs of request/reply queues are created for this test. One or more requester applications will send messages to one of the application request queues and will wait for a reply on the associated reply queue. Responder applications will listen for messages on the request queues before sending them to the correct reply queue.

Subsequent requester applications will send and receive messages from the set of application queues on a round-robin basis i.e. distributing the messages produced and consumed across the set of application queues (the diagram below shows how the distribution would cycle round, if only three queue pairs are used).

Each test is scaled up by adding additional batches of requesters in stages, until the limiting factor is reached. Depending on the nature of the test (message size, latency of the file system hosting the transaction log etc) the number of requesters added before the limit of the test is reached, will differ.

MQ-CPH was used as the test application.

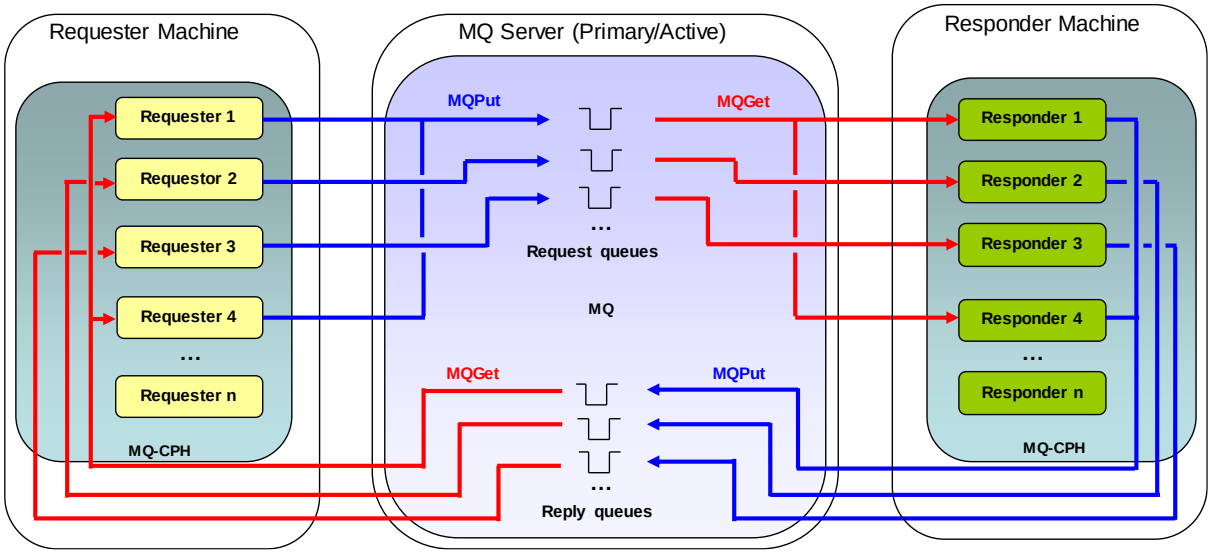

FIGURE 7 - REQUESTER-RESPONDER TEST SCENARIO

Test Scenario 2 – Putter/Getter (Persistent messages)

For the fail-over tests, a Putter/Getter test scenario was used. Ten queues are created for this test.

One or more Put/Get applications will send messages to one of the application request queues and will then Get the message back off the same queue. Subsequent Put/Get applications will send and receive messages from the set of application queues on a round-robin basis i.e. distributing the messages produced and consumed across the set

of application queues (the diagram below shows how the distribution would cycle round, if only three queues are used).

A modified version of JMSPerfharness was used (modified to attempt reconnection to a secondary QM, when a JMS API call to the primary QM returned with an error).

In contrast to the Requester/Responder scenario, this scenario was rated, i.e. the Put/Get applications were set to execute n loops/sec, maintaining steady rate of about 50,000/sec where this was achievable. This capped the MQ server CPU to around 70%, in most cases, so that fail-over was not occurring when the machine was already CPU constrained by the workload running as fast as it could.

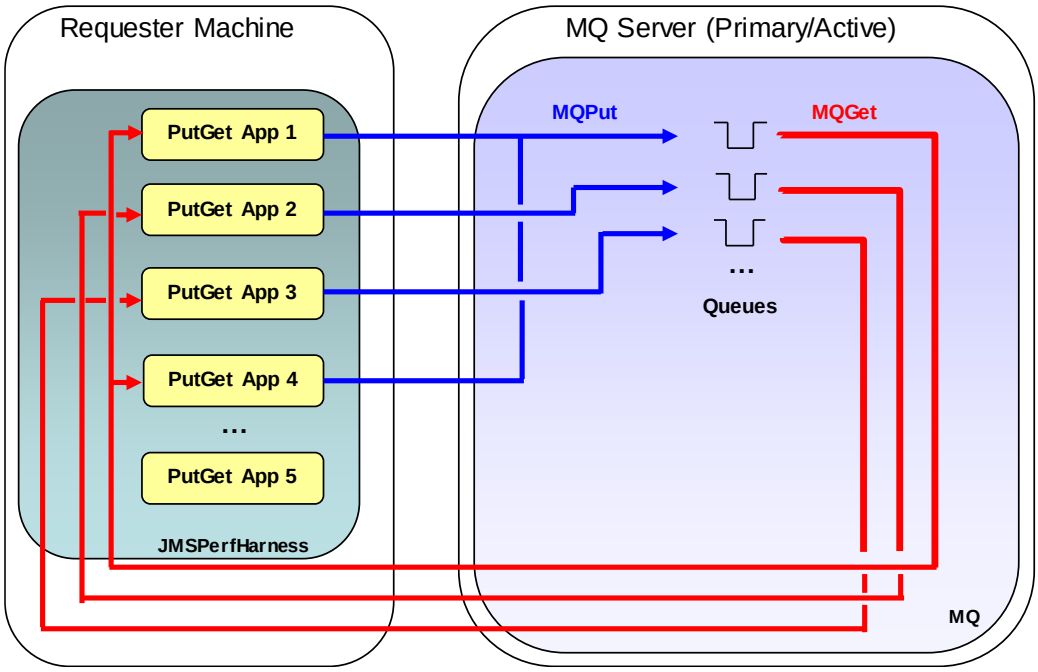

FIGURE 8 - PUTTER/GETTER TEST SCENARIO

Appendix C : Utilities

RDQM Related Commands

For RDQM, the HA current location, and status of the RDQM nodes can be viewed using **rdqmstatus**. You can also monitor the Pacemaker cluster, using **crm_mon**.

Running rdqmstatus commands from a node hosting a secondary QM in the RDQM HA group (mqperfx2), in a healthy Pacemaker cluster, where the primary QM is currently running on the node (mqperfxs) will return the following output, for example:

```
[mqperf@mqperfx2 ~]$ rdqmstatus
Node:                                   mqperfx2

Queue manager name:                     PERF0
Queue manager status:                   Running elsewhere
HA current location:                    mqperfxs
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[mqperf@mqperfx2 ~]$ rdqmstatus -n
Node mqperfxs is online
Node mqperfxw is online
Node mqperfx2 is online
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

Running the crm_mon command from another node in the cluster (mqperfx2) when the Pacemaker cluster is in the same state, will return the following output, for example:

```
[mqperf@mqperfx2 ~]$ crm_mon
Stack: corosync
Current DC: mqperfxw (version 1.1.15.linbit-2.0+20160622+e174ec8.el7-e174ec8) - partition
with quorum

Last updated: Wed Jun  6 10:17:29 2018          Last change: Wed Jun  6 10:17:26 2018 by
mqm via crm_attribute on mqperfxs

3 nodes and 6 resources configured

Online: [ mqperfx2 mqperfxs mqperfxw ]

 Master/Slave Set: ms_drbd_perf0 [p_drbd_perf0]
     Masters: [ mqperfxs ]
     Slaves: [ mqperfx2 mqperfxw ]

p_fs_perf0     (ocf::heartbeat:Filesystem):    Started mqperfxs
p_rdqmx_perf0   (ocf::ibm:rdqmx):     Started mqperfxs
perf0   (ocf::ibm:rdqm):        Started mqperfxs
```

Simulating Network latency (tc)

Latencies were injected into the network interfaces where applicable, using tc (traffic control).

The example below, shows how to set a 500us latency on interface ens1f0, the 10Gb link.

```
[root@mqperfxs mqperf]# ping mqperfx2
PING mqperfx2.hursley.ibm.com (9.20.36.121) 56(84) bytes of data.
64 bytes from mqperfx2.hursley.ibm.com (9.20.36.121): icmp_seq=1 ttl=64 time=0.181 ms
```

```
64 bytes from mqperfx2.hursley.ibm.com (9.20.36.121): icmp_seq=2 ttl=64 time=0.090 ms
^C

--- mqperfx2.hursley.ibm.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.090/0.135/0.181/0.046 ms

[root@mqperfxs mqperf]# tc qdisc add dev ens1f0 root netem delay 500us
[root@mqperfxs mqperf]# tc qdisc show dev ens1f0
qdisc netem 8001: root refcnt 33 limit 1000 delay 499us

[root@mqperfxs mqperf]# ping mqperfx210
PING mqperfx210.hursley.ibm.com (10.20.36.121) 56(84) bytes of data.
64 bytes from mqperfx210.hursley.ibm.com (10.20.36.121): icmp_seq=1 ttl=64 time=0.531 ms
64 bytes from mqperfx210.hursley.ibm.com (10.20.36.121): icmp_seq=2 ttl=64 time=0.532 ms
^C

--- mqperfx210.hursley.ibm.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.531/0.531/0.532/0.023 ms

[root@mqperfxs mqperf]# tc qdisc delete dev ens1f0 root
```

Simulating Network outage (iptables)

In order to simulate a network outage on the primary queue manager, the iptables command was used.

NB: Do not use '**ifdown'** command to simulate an outage. This will disable the device and remove the IP address so that DRBD can no longer bind to it. The use of this command is not suitable for fail-over testing.

The iptables commands used are shown below, being used to drop all inbound and outbound packets on interfaces ens3 & ens1f0 (the 10Gb and 40Gb links). The subsequent list-rules format of the command shows the new rules.

```
iptables -A INPUT -i ens3 -j DROP
iptables -A OUTPUT -o ens3 -j DROP
iptables -A INPUT -i ens1f0 -j DROP
iptables -A OUTPUT -o ens1f0 -j DROP
iptables -S

-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i ens3 -j DROP
-A INPUT -i ens1f0 -j DROP
-A OUTPUT -o ens3 -j DROP
-A OUTPUT -o ens1f0 -j DROP
```

Deleting the rules will stop packets being dropped, e.g.:

```
iptables -D INPUT -i ens3 -j DROP
iptables -D OUTPUT -o ens3 -j DROP
iptables -D INPUT -i ens1f0 -j DROP
iptables -D OUTPUT -o ens1f0 -j DROP
```

Appendix D : Glossary of terms used in this report

CD	Continuous delivery.
DRBD	Distributed, replicated block device.
HA	High availability.
JMSPerfharness	JMS based, performance test application (https://github.com/ot4i/perf-harness)
LTS	Long term service.
MIQM	Multi-instance queue manager.
MQ-CPH	C based, performance test application (https://github.com/ibm-messaging/mq-cph)
RDQM	Replicated data queue manager.

Appendix E : Additional Resources

There is a wealth of information on RDQM in the MQ V9 Knowledge Centre, which you should refer to, but the following additional resources can be helpful.

RDQM (Easy HA) - Getting started
https://youtu.be/5qYHsmKZt2M

RDQM in MQ Advanced 9.0.4
https://developer.ibm.com/messaging/2017/10/25/rdqm-mq-advanced-9-0-4

IBM MQ: How long will it take to (re)start my queue manager?
https://developer.ibm.com/messaging/2017/10/25/qm_restart_time

RDQM GitHub
https://github.com/ibm-messaging/mq-rdqm

MQ-CPH (The IBM MQ C Performance Harness)
https://github.com/ibm-messaging/mq-cph

JMSPerfHarness
https://github.com/ot4i/perf-harness