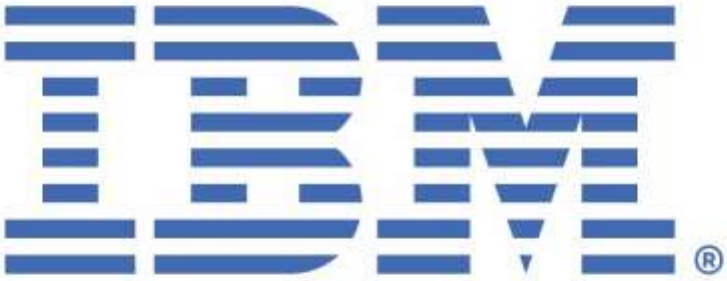


IBM MQ HA (NHA & RDQM)

Performance Report (x86-64 platform)

Version 1.0 - February 2026

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

Notices

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, January 2026.

This edition applies to *IBM MQ V9.4.5* (and to all subsequent releases and modifications until otherwise indicated in new editions).

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 V9.4. The information is not intended as the specification of any programming interface that is provided by IBM MQ. It is assumed that the reader is familiar with the concepts and operation of IBM MQ V9.4.

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.

Preface

Target audience

The report is designed for people who:

- Want to understand the performance implications of NHA and RDQM HA solutions in MQ v9.4.5 for Linux on x86_64.

Note that this report does not include NHA-CRR or RDQM DR data.

The reader should have a general awareness of the Linux operating system and of IBM MQ HA solutions (for NHA and RDQM) to make best use of this report.

Whilst operating system, and MQ tuning details are given in this report (specific to the workloads presented), a more general consideration of tuning and best practices is available in a separate paper here:

https://ibm-messaging.github.io/mqperf/MQ_Performance_Best_Practices_v1.0.1.pdf

Feedback

We welcome feedback on this report.

- Does it provide the sort of information you want?
- Do you feel something important is missing?
- Is there too much technical detail, or not enough?
- Could the material be presented in a more useful manner?

Specific queries about performance problems on your IBM MQ system should be directed to your local IBM Representative or Support Centre.

Please direct any feedback on this report to paul_harris@uk.ibm.com.

Contents

TABLES

FIGURES

1 Introduction

The V9.4.4 and V9.4.5 continuous delivery (CD) release of MQ included new high availability (HA) capabilities to the advanced edition:

V9.4.4:

- Native High Availability (NHA) and Cross Region Replication (NHA-CRR) on Linux
- TLS Secured replication links in your HA RDQM, DR RDQM, and DR/HA RDQM configuration.

See: https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=delivery-whats-new-changed-in-mq-944

V9.4.5:

- Secure heartbeat for HA RDQM (You can specify that the heartbeat links in your HA RDQM configuration are encrypted.)

See: https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=delivery-whats-new-changed-in-mq-945

High availability for MQ typically minimises the time that a queue manager (and its messages) is not available to applications, and removes single points of failure in the infrastructure where MQ runs. This typically requires:

- Fast detection of a failure of the running queue manager, and automatic recovery, often to a second physical system
- Persisting queue manager data to multiple physical disks

There are a number of options available to achieve high availability in MQ, including setting up general high availability managers, or deploying on MQ Appliances (see High availability configurations in the MQ doc).

This report gives a performance overview of using the two quorum-based HA capabilities of MQ (NHA and RDQM) on Linux x64 hosts using Red Hat Enterprise Level 9.7.

Other HA solutions are available for use with MQ: https://developer.ibm.com/articles/mq-ha-dr-options

As with all performance sensitive tests, you should run your own tests where possible, to simulate your production environment and circumstances you are catering for.

This report does not include NHA-CRR or RDQM DR data.

2 Base MQ Performance Workloads

A single style of persistent messaging workload was used for all the measurements in this report. This is a requester/responder (RR) scenario which is synchronous in style because the application putting a message on a queue will wait for a response on the reply queue before putting the next message. This workload typically runs 'unrated' (no think time between getting a reply and putting the next message on the request queue).

The tests are run using 'client mode' applications where requesters, and responders are on separate hosts to the MQ (see description of the 'RR-CC' workload in the next section).

Client mode connections use fastpath channels and listeners (trusted) and have SHARECNV set to 1, which is the recommended value for performance.

Applications, Threads and Processes

From a queue manager's perspective in the workloads described below, each connection represents a unique application. The workloads are driven by the MQ-CPH or Perfharness client emulator tools. Both these tools are multi-threaded so 10 applications may be represented by 10 threads within a single MQ-CPH process, for instance. If 200 responder applications are started, this will always be represented by 200 threads, but they could be spread across 10 processes (each with 20 threads). The main point is that each application below is a single thread of execution within MQ-CPH or JMSPerfHarness, spread across as many processes as makes sense.

2.1 The RR-CC Workload

(Client mode requesters with client mode responders.)

Please note that this is a simplified diagram which doesn't show the HA topology, only the active QM that is currently serving applications.

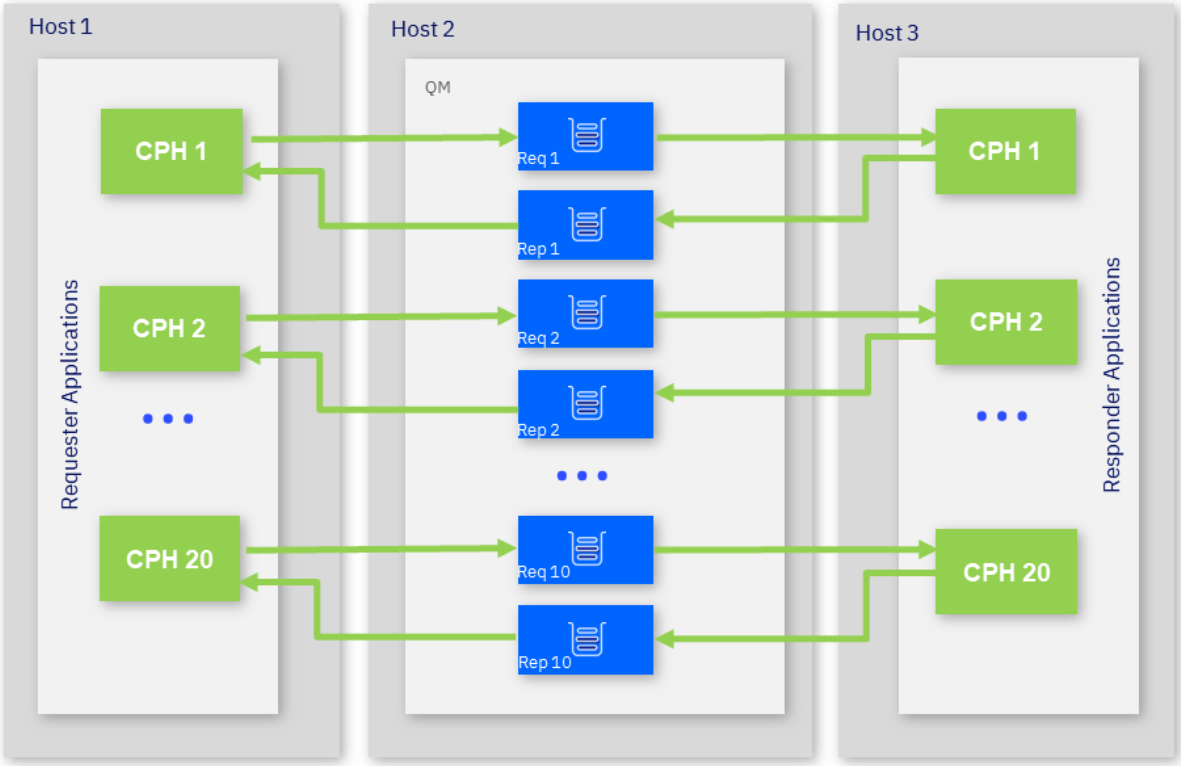


FIGURE 1 – TEST APPLICATION TOPOLOGY

Figure 1 shows the topology of the RR-CC test. The test simulates multiple 'requester' applications which all put messages onto a set of ten request queues. Additional machines may be used to drive the requester applications where necessary.

Another set of 'responder' applications retrieve the message from the request queue and put a reply of the same length onto a set of ten reply queues. The number of responders is set such that there is always a waiting 'getter' for the request queue.

The applications utilise the requester and responder queues in a round robin fashion, ensuring even distribution of traffic, so that in the diagram above CPH11 will wrap round to use the Rep1/Req1 queues, and CPH 20 will use the Req10/Rep10 queues.

The flow of the test is as follows:

- The requester application puts a message to a request queue on the remote queue manager and holds on to the message identifier returned in the message descriptor. The requester application then waits indefinitely for a reply to arrive on the appropriate reply queue.
- The responder application gets messages from the request queue and places a reply to the appropriate reply queue. The queue manager copies over the message

identifier from the request message to the correlation identifier of the reply message.

- The requester application gets a reply from the reply queue using the message identifier held when the request message was put to the request queue, as the correlation identifier in the message descriptor.

This test is executed using client channels as trusted applications by specifying "*MQIBindType=FASTPATH*" in the qm.ini file. This is recommended generally but not advised if you run channel exit programs and do not have a high degree of confidence in their robustness.

Network Flows:

As the topology utilises separate hosts for the requesters and responders, each round trip will comprise of 2 inbound messages to the server and 2 outbound messages from the server, all being transmitted across the network. So, if the message size is 2048 bytes there will be 2 x (2048 + metadata) inbound to the MQ server and 2 x (2048 + metadata) outbound from the server, where metadata is the non-message payload data, comprising of the MQ and transport headers.

TLS Security:

All the workloads in this report used TLS enabled clients negotiating with the TLS12_ANY option.

The replications links are unsecured unless otherwise specified.

Message Rates:

All rates in this report are specified as round-trips/sec. As each round-trip comprises of a request message and a reply message the number of messages/sec is 2 x the round-trip rate i.e.:

1,000 round-trips/sec = 2,000 messages/sec processed by the queue manager.

3 Replicated Logging

Native HA relies on a recovery log type of *replicated*[a]. This is conceptually similar to Linear logging with automatic log management. The NHA technology only replicates data in the recovery log, and dynamically builds the wider set of queue manager data files from that information. As such there needs to be a regular point of consistency to recover from, so media images are taken of the QM state when required. These are included in the recovery log data sent to the HA replica queue managers.

RDQM replicates all data that is persisted for a queue manager, including the recovery logs and the queue files. Therefore, it does not require the same media images used by NHA. RDQM tests used circular logging.

3.1 Replicated Logging Performance and 'Warmup'

As a performance optimisation, replicated logging efficiently reuses preexisting log extents. For a newly created queue manager, optimal performance is achieved once log extent reuse is occurring rather than the initial creation of new log extents. This occurs once:

1. The queue manager has processed enough messages to have caused a number of log extents to have been created and formatted.
2. A media image has been taken, moving the point of recovery forward and making old log extents available for re-use (you can see these in the MQ active log directory as log extents whose filenames are prefixed with 'R'.

When a media image is taken will depend on two attributes of the queue manager:

IMGINTVL Specifies a target frequency (in minutes) for recording media images.

IMGLOGLN Specifies the target amount (in megabytes) of log written after which media images are taken.

Native HA queue managers are created with IMGLOGLN set to 25% of the available space on the volume where the recovery logs are to be written, but as we had very large filesystems for the tests run in this report (which would have resulted in a value of 256GB for IMGLOGLN) that value was reduced to 100GB.

IMGINTVL was set to 60 minutes for all tests which means it was not a factor in deciding on when image copies are taken (all the tests here are <60 minutes in duration).

When a media image is taken, log extents freed up can be reused or deleted. MQ will decide on the number of log extents to reuse or delete depending on the workload, but as our workloads increasingly load the queue manager, MQ may delete log extents that would benefit us keeping with the increasing delivery rate of the test.

To ensure we keep an optimal number of extents, the qm.ini parm **MinReusableLogExtents** was set to 750.

[a] See https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=nh-performance-considerations-when-moving-from-circular-linear-logs

Prior to each test in this report, we ran the queue manager for a sufficient period of time so that a media image was taken and log extent reuse was occurring.

Re-useable logs can be seen in the filesystem as active log files prefixed with ‘R’. Having these in the active log directory is a good thing as they are log extents that have already been formatted and freed up, ready for re-use, e.g. (truncated output):

```
[user@mqhost active]$ pwd
/var/mqm/log/QM1/active
[user@mqhost active]$ ls
R0001305.LOG  R0001420.LOG  R0001535.LOG  S0001650.LOG  S0001765.LOG  S0001880.LOG  S0001995.LOG
R0001306.LOG  R0001421.LOG  R0001536.LOG  S0001651.LOG  S0001766.LOG  S0001881.LOG  S0001996.LOG
R0001307.LOG  R0001422.LOG  R0001537.LOG  S0001652.LOG  S0001767.LOG  S0001882.LOG  S0001997.LOG
…
```

3.2 Reporting Log Extent Reuse in the MQ Error Log

MQ emits two useful messages to the queue manager error log that that can help you understand how log files are being reused and when media images that release older log extents are taken (not all media images will release older log extents).

AMQ7490I Log extent statistics message that contains information on the number of log extents created, reused or deleted since this message was previously issued.

AMQ7468I The oldest log file required to perform media recovery of queue manager

A utility is available on the mqperf github site[b] to produce a summary from those messages. E.g. from the error log for the warmup prior to the first test run in this report, the following data was reported (truncated):

```
Parsing file /var/mqm/qmgrs/PERF0/errors/AMQERR01.LOG
26/11/25 14:37:07 ===> Media image has updated media recovery file to S0001562.LOG.
26/11/25 14:37:07      created:404     reused:32       deleted:0       Restart log: S0001996.LOG.
26/11/25 14:37:45      created:30      reused:0        deleted:0       Restart log: S0001996.LOG.
26/11/25 14:37:46      created:5       reused:0        deleted:0       Restart log: S0002002.LOG.
26/11/25 14:37:47      created:6       reused:0        deleted:0       Restart log: S0002008.LOG.
...
26/11/25 14:38:19      created:5       reused:0        deleted:0       Restart log: S0002320.LOG.
26/11/25 14:38:19      created:4       reused:0        deleted:0       Restart log: S0002324.LOG.
26/11/25 14:38:20      created:6       reused:0        deleted:0       Restart log: S0002330.LOG.
26/11/25 14:38:20 ===> Media image has updated media recovery file to S0001563.LOG.
26/11/25 14:38:20      created:5       reused:0        deleted:0       Restart log: S0002335.LOG.
26/11/25 14:38:21 ===> Media image has updated media recovery file to S0001953.LOG.
26/11/25 14:38:21      created:1       reused:3        deleted:0       Restart log: S0002339.LOG.
26/11/25 14:38:21      created:0       reused:5        deleted:0       Restart log: S0002344.LOG.
26/11/25 14:38:22      created:0       reused:6        deleted:0       Restart log: S0002350.LOG.
26/11/25 14:38:22      created:0       reused:6        deleted:0       Restart log: S0002356.LOG.
26/11/25 14:38:23      created:0       reused:7        deleted:0       Restart log: S0002363.LOG.
...
26/11/25 14:40:46      created:0       reused:1        deleted:0       Restart log: S0003787.LOG.

No log extents deleted
Totals -               created:776     reused:1480     deleted:0
```

In the output above, older log extents were released by the media images taken at 14:38:20 and 14:38:21 and log extent reuse started to occur. Note that further log extents may need to be created in the future, depending on the workload, but the initiation of log reuse should improve performance of linear logging moving forward.

[b] https://github.com/ibm-messaging/mqperf/blob/gh-pages/samp/linear_log_stats.py

4 NHA Replication Results.

The following section details results of running the RR-CC workload with 2KB, 20KB and 200KB message sizes, comparing a single instance queue manager, with no resilience to storage failures, and an NHA configuration (HA replication only).

In these tests, to highlight the NHA behaviour, especially as data levels increase, the difference between the single instance queue manager and the NHA queue manager has been accentuated through the use of very fast, locally attached, storage. In this scenario, the data replication across the network, required by all multi-node resilient technologies, becomes an unavoidable expense of achieving high availability.

The topologies for the test are shown in 0.

The scenarios shown are:

Scenario	Description
Base	Single instance, non-HA queue manager.
NHA	Three QM NHA-HA topology with unsecured replication links.
NHA Crypt	Three QM NHA-HA topology with TLS secured replication links (TLS12_ANY)

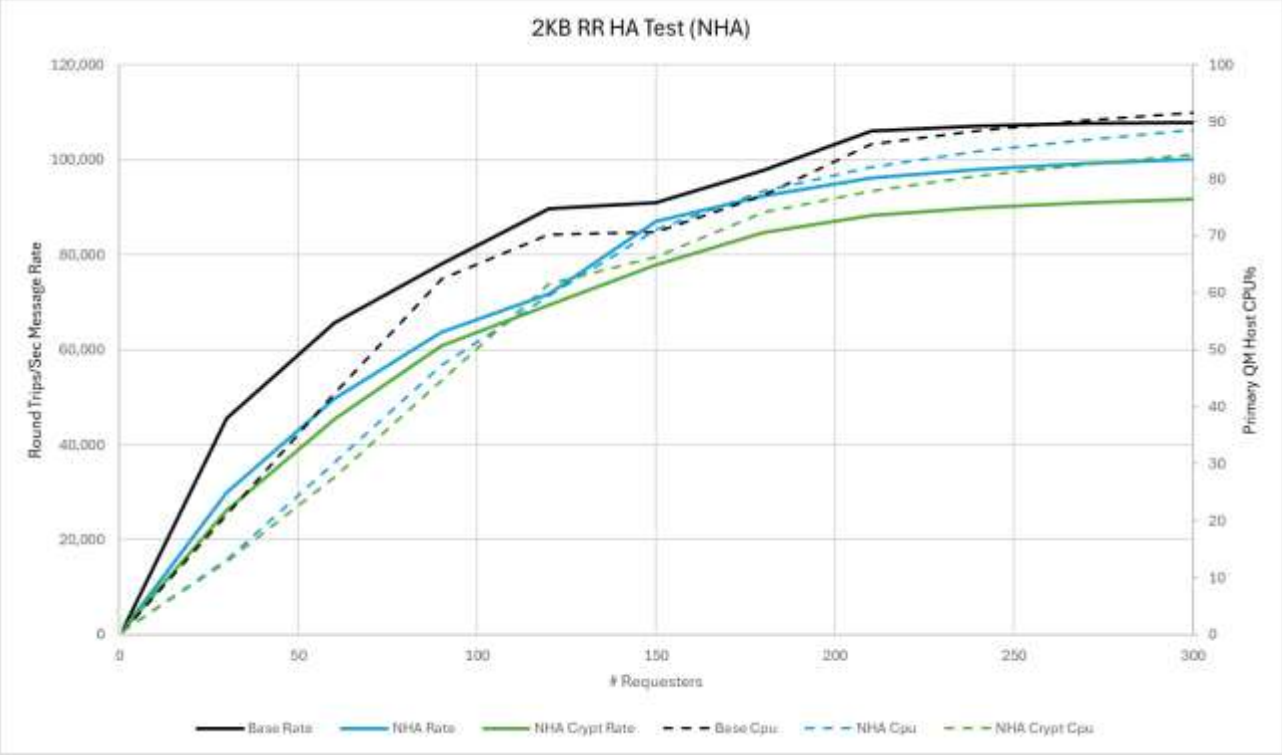


FIGURE 2 2KB RR NHA(HA) TEST RESULTS

For the 2KB message size there is a relatively low overhead using NHA with local, high bandwidth network links (all these tests utilise 100Gb replication links with a ping time of <50µs).

Enabling encryption introduces a further overhead as expected. Your choice of cipher may have a different impact, which should be tested.

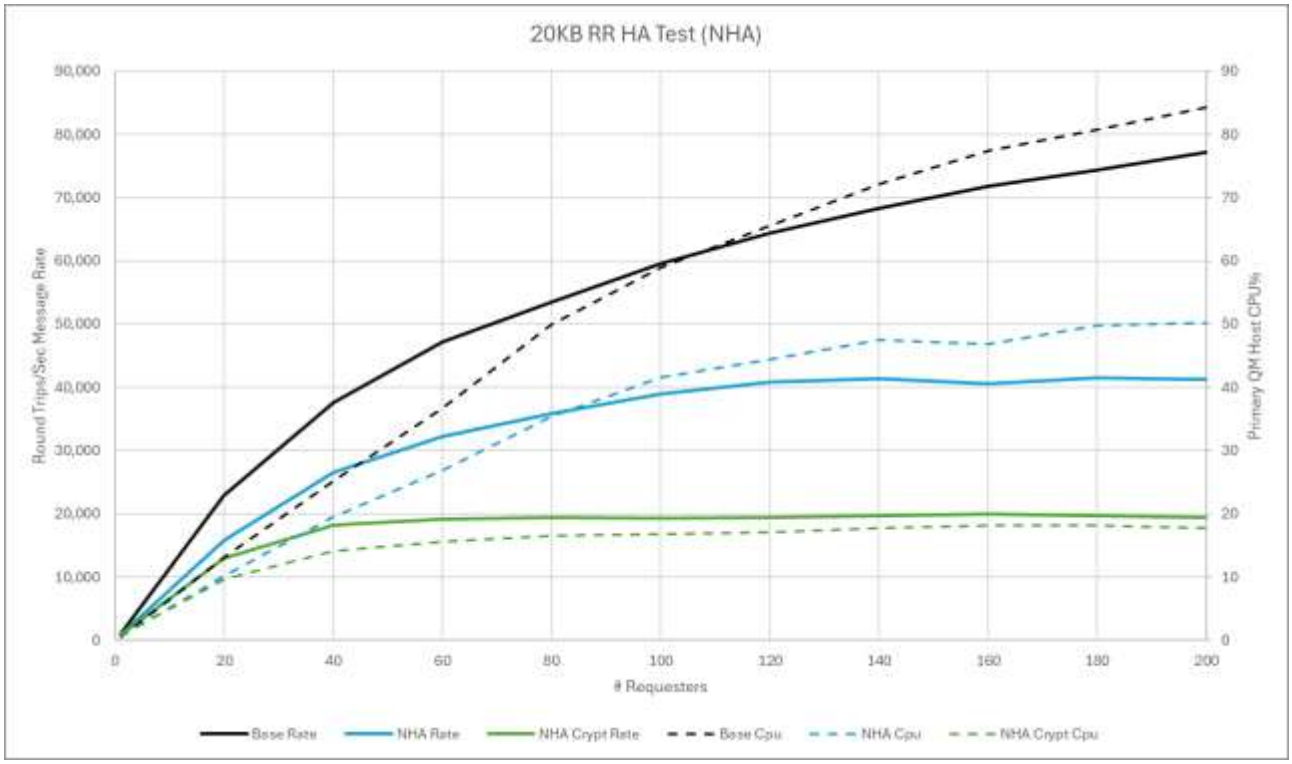


FIGURE 3 20KB RR NHA(HA) TEST RESULTS

With a 20KB message size, 10x more data is being written to the recovery log, which needs to be replicated, so the impact is more substantial. Encryption is also more costly as the data encryption path is a greater proportion of the total pathlength as the message size increases.

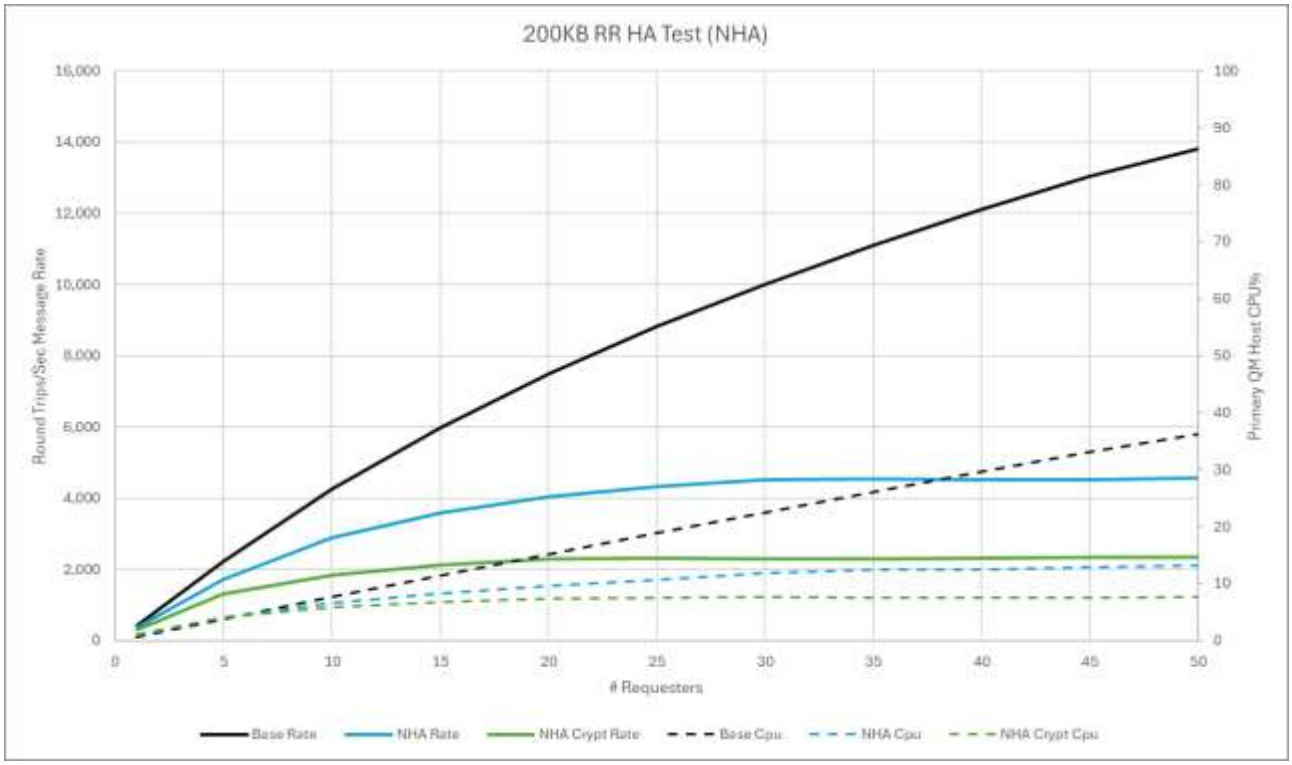


FIGURE 4 200KB RR NHA(HA) TEST RESULTS

With 200KB messages the overhead increases further, whilst encryption halves the NHA non-encrypted rate (similar to the 20KB case).

This difference is accentuated given the fast NVMe disks used locally. It would be expected that if you are using slower disks (or SAN storage) then HA replication overhead would be less significant.

All the tests above stress the test environment to the limit. In your own environment you may see different results depending on the performance characteristics of the network links and of the filesystems hosting the recovery log. Messaging rates in production may well be towards the left-hand end of these graphs where the impact is proportionally lower.

4.1 Impact of Replication Link Quality on Performance

Each of the three HA instances of the NHA queue managers will replicate recovery log data over the replication links. In the case of the tests in the previous section these are 100Gb links with low latency (<50µ seconds ping time when not under load), representing a deployment across very closely connected nodes.

To ensure message consistency and zero data loss, HA replication requires a level of synchronisation across nodes, so the quality of the replication links will have a direct impact on performance (as we need to ensure data is committed to at least one HA instance with each write to the local recovery log). Therefore, as you increase the distance between nodes, or reduce the quality of their connectivity, typically there will be change in the topline throughput of the queue manager, dependant on how synchronous the application usage pattern is and how large the recovery log writes are (larger log writes will optimise the bandwidth utilisation to the recovery log)

Our test scenario, each requestor application requires a message to complete a round-trip (onto one queue, from that queue, on to a second queue and off again) before the next message is sent. This means that any latency in the replication link will have a significant impact on the individual applications in the test, and this will ultimately impact the overall topline throughput of the test. E.g. if we introduce a 2ms round-trip delay in the replication links, throughput will drop (see Figure 5).

Messaging patterns that are more asynchronous in their end-to-end nature may see a different impact as network latency increases.

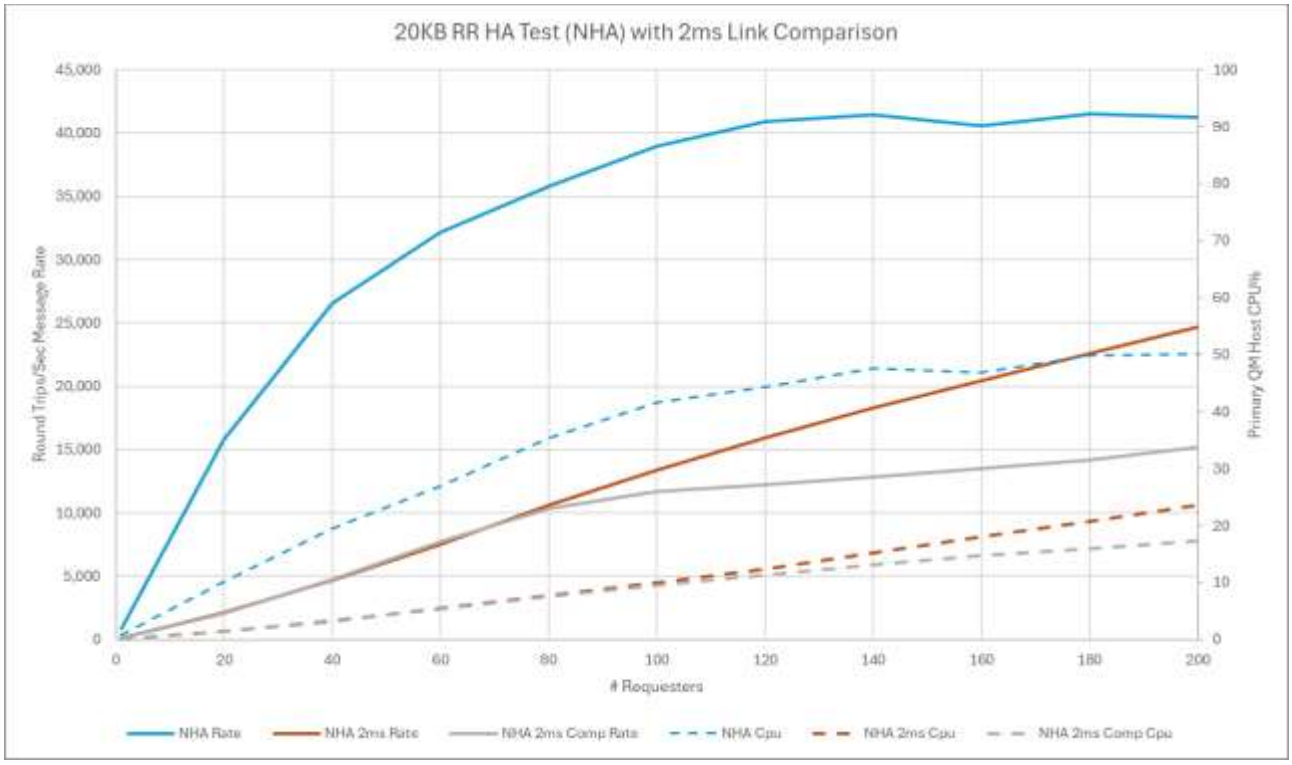


FIGURE 5 20KB RR NHA(HA) TEST RESULTS WITH 2MS DELAY ON REPLICATION LINKS

With a 2ms latency added to the link, the turnaround time for recovery log replication was slower, significantly reducing the maximum throughput in this round-trip messaging scenario. This illustrates the necessity to test the infrastructure you deploy any HA solution on, and test with representative application patterns. HA replication will only be as fast as the slowest path through which recovery log data is persisted, including the replication link.

Note that the as the number of clients increases, the concurrency of the workload means that log writes are larger, lessening the effect of the higher latency replication links.

4.2 NHA Diagnostics

You can use the sample system statistics tool to report metrics for NHA, e.g.

```
/opt/mqm/samp/bin/amqsrua -m PERF0 -c NHAREPLICA -t REPLICATION -o +
```

This will print out the system statistics for both of the NHA replicas (every 10seconds by default). E.g. if the replica hosts for queue manager ‘QM1’ are live_host2 and live_host3, the following output will be displayed:

```
Publication received PutDate:20251223 PutTime:11491557 Interval:10.000 seconds
live_host2QM1            Synchronous log bytes sent 85160310 8515839/sec
live_host2QM1            Synchronous compressed log bytes sent 0
live_host2QM1            Synchronous log data average compression time 0 uSec
live_host2QM1            Synchronous uncompressed log bytes sent 85160310 8515839/sec
live_host2QM1            Catch-up log bytes sent 0
live_host2QM1            Catch-up compressed log bytes sent 0
live_host2QM1            Catch-up log data average compression time 0 uSec
live_host2QM1            Catch-up uncompressed log bytes sent 0
live_host2QM1            Log write average acknowledgement latency 80 uSec
live_host2QM1            Log write average acknowledgement size 2256
live_host2QM1            Backlog bytes 2988
live_host2QM1            Backlog average bytes 150154
live_host2QM1            Average network round trip time 56 uSec
live_host2QM1            Synchronous log bytes decompressed 0
live_host2QM1            Synchronous log data average decompression time 0 uSec
live_host2QM1            Catch-up log bytes decompressed 0
live_host2QM1            Catch-up log data average decompression time 0 uSec
live_host2QM1            Acknowledged log sequence number <0:0:4932:56457> 0x000000001344dc89
live_host2QM1            Log file system - bytes in use 84716568576
live_host2QM1            Log file system - free space 94.77%
live_host2QM1            Queue Manager file system - bytes in use 80792MB
live_host2QM1            Queue Manager file system - free space 94.77%
live_host2QM1            MQ FDC file count 0
live_host2QM1            Log - write latency 11 uSec
live_host2QM1            Log - write size 5632
live_host2QM1            Log - slowest write since restart 253379 uSec
live_host2QM1            Log - timestamp of slowest write 1766490500428657 uSec
```

```
Publication received PutDate:20251223 PutTime:11491557 Interval:10.000 seconds
live_host3QM1            Synchronous log bytes sent 85160310 8515839/sec
live_host3QM1            Synchronous compressed log bytes sent 0
live_host3QM1            Synchronous log data average compression time 0 uSec
live_host3QM1            Synchronous uncompressed log bytes sent 85160310 8515839/sec
live_host3QM1            Catch-up log bytes sent 0
live_host3QM1            Catch-up compressed log bytes sent 0
live_host3QM1            Catch-up log data average compression time 0 uSec
live_host3QM1            Catch-up uncompressed log bytes sent 0
live_host3QM1            Log write average acknowledgement latency 74 uSec
live_host3QM1            Log write average acknowledgement size 2256
live_host3QM1            Backlog bytes 2988
live_host3QM1            Backlog average bytes 150154
live_host3QM1            Average network round trip time 53 uSec
live_host3QM1            Synchronous log bytes decompressed 0
live_host3QM1            Synchronous log data average decompression time 0 uSec
live_host3QM1            Catch-up log bytes decompressed 0
live_host3QM1            Catch-up log data average decompression time 0 uSec
live_host3QM1            Acknowledged log sequence number <0:0:4932:56457> 0x000000001344dc89
live_host3QM1            Log file system - bytes in use 84716564480
live_host3QM1            Log file system - free space 94.77%
live_host3QM1            Queue Manager file system - bytes in use 80792MB
live_host3QM1            Queue Manager file system - free space 94.77%
live_host3QM1            MQ FDC file count 0
live_host3QM1            Log - write latency 11 uSec
live_host3QM1            Log - write size 5632
live_host3QM1            Log - slowest write since restart 272812 uSec
live_host3QM1            Log - timestamp of slowest write 1766490510609894 uSec
```

These statistics can give you an insight into where a bottleneck may be. Network times are given and the write times to the recovery logs on the replica machines.

5 RDQM Replication Results.

RDQM provides a similar level of message replication and automatic recovery as Native HA. However, it uses different internal technology and techniques to achieve this. Just as we did for NHA, the following section details results of running the RR-CC workload with 2KB, 20KB and 200KB message sizes, comparing a single instance queue manager and an RDQM configuration (HA replication only). The topologies for the test are shown in 0.

The scenarios shown are:

Scenario	Description
Base	Single instance non-HA queue manager.
RDQM HA	Three QM RDQM-HA topology with unsecured replication links.
RDQM Crypt	Three QM RDQM-HA topology with TLS secured replication links.

As previously described, RDQM queue managers can be configured to use circular logging, while meeting the same high availability characteristics as NHA. The performance of circular logging is similar to linear logging in general (once linear logs are being reused) but circular logging does not require media images to be written to the recovery logs.

For all the results below, the initial replication of the filesystem was completed, i.e. each node shown in the 'rdqmstatus' command reported:

HA status: **Normal**

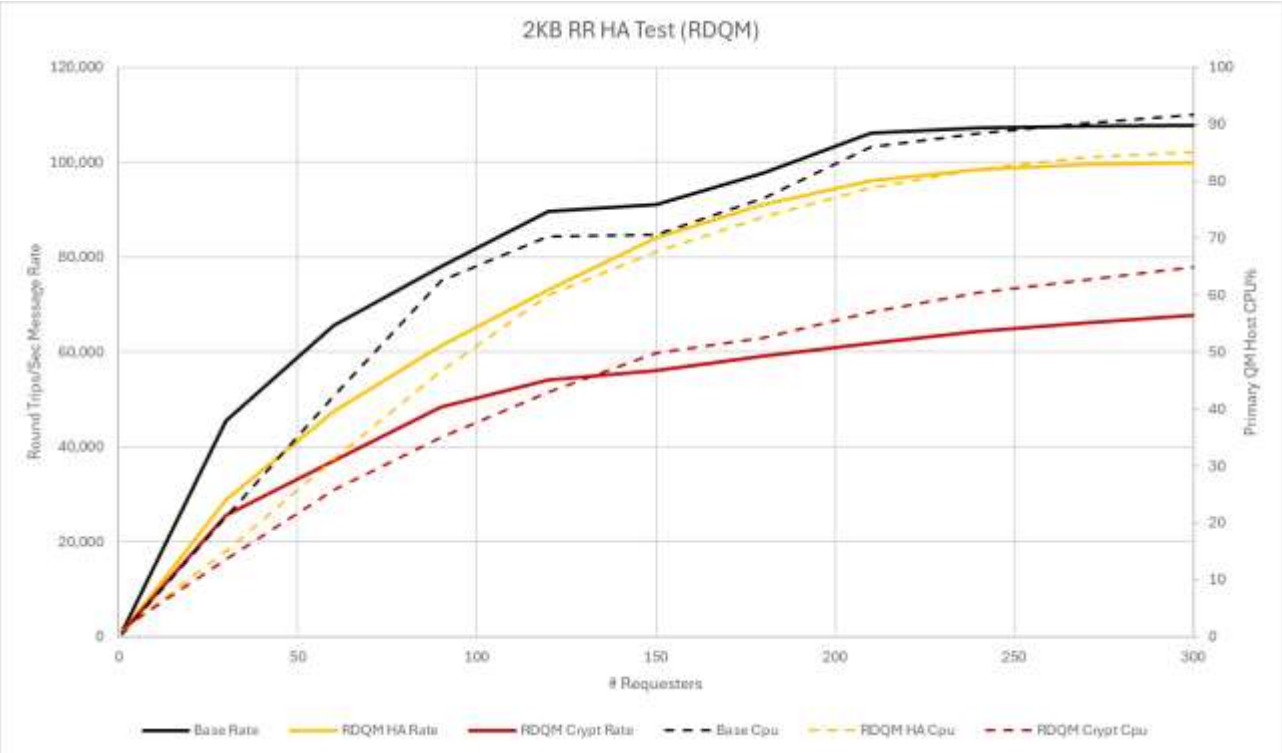


FIGURE 6 2KB RR RDQM(HA) TEST RESULTS

Results are similar to the NHA results for 2K with overhead increasing when encryption is enabled.

Remember that these tests utilise 100Gb replication links with a ping time of 25µs and that your choice of encryption cipher will have some impact on performance.

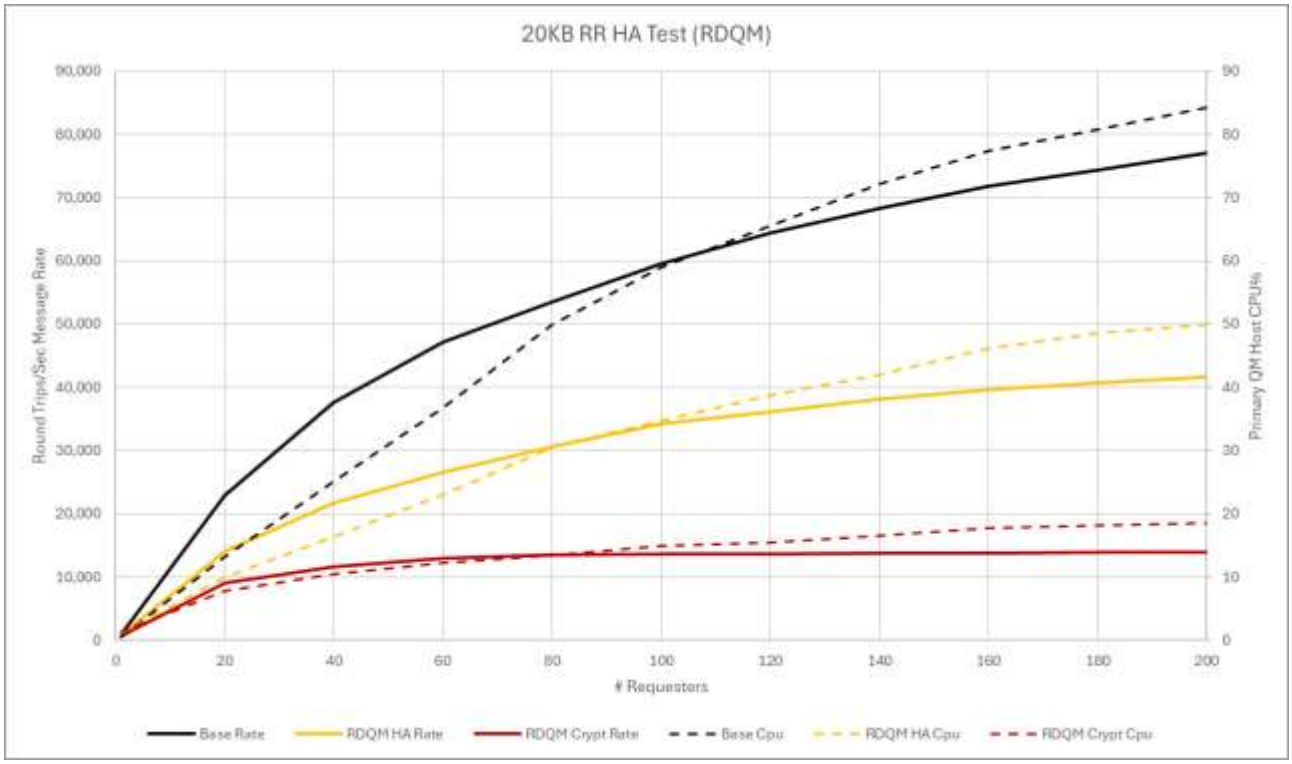


FIGURE 7 20KB RR RDQM(HA) TEST RESULTS

As the message sizes increases so replication has a bigger impact and encryption of the data will account for a higher proportion of the pathlength as seen in the NHA tests.

FIGURE 8 200KB RR RDQM(HA) TEST RESULTS

As for NHA the impact of replication is more evident with larger message sizes. This difference is accentuated given the fast NVMe disks used locally. It would be expected that if you are using slower disks (or SAN storage) then HA replication overhead would be less significant.

5.1 Secure heartbeat for HA RDQM

With MQ V9.4.5 you now have the option to encrypt the RDQM heartbeat. This was tested with the RDQM HA runs and no noticeable performance impact was registered on messaging throughput, though you should test in your own environment.

See: https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=availability-secure-heartbeat-ha-rdqm

5.2 Impact of Replication Link Quality

Just as for NHA, the RDQM tests in the previous section were setup so that replication between the primary and secondary nodes utilised dedicated 100Gb links with low latency (<50µ seconds ping time when not under load). These are the same link used to replicate recovery log data in the NHA tests.

Similar to NHA, RDQM HA replication is synchronous in nature so the quality of the replication links will have a direct impact on performance (we need to ensure that committed data is written to both the secondary nodes before continuing). E.g. if we introduce a 2ms round-trip delay in the replication links, throughput will drop in certain messaging scenarios (see Figure 9).

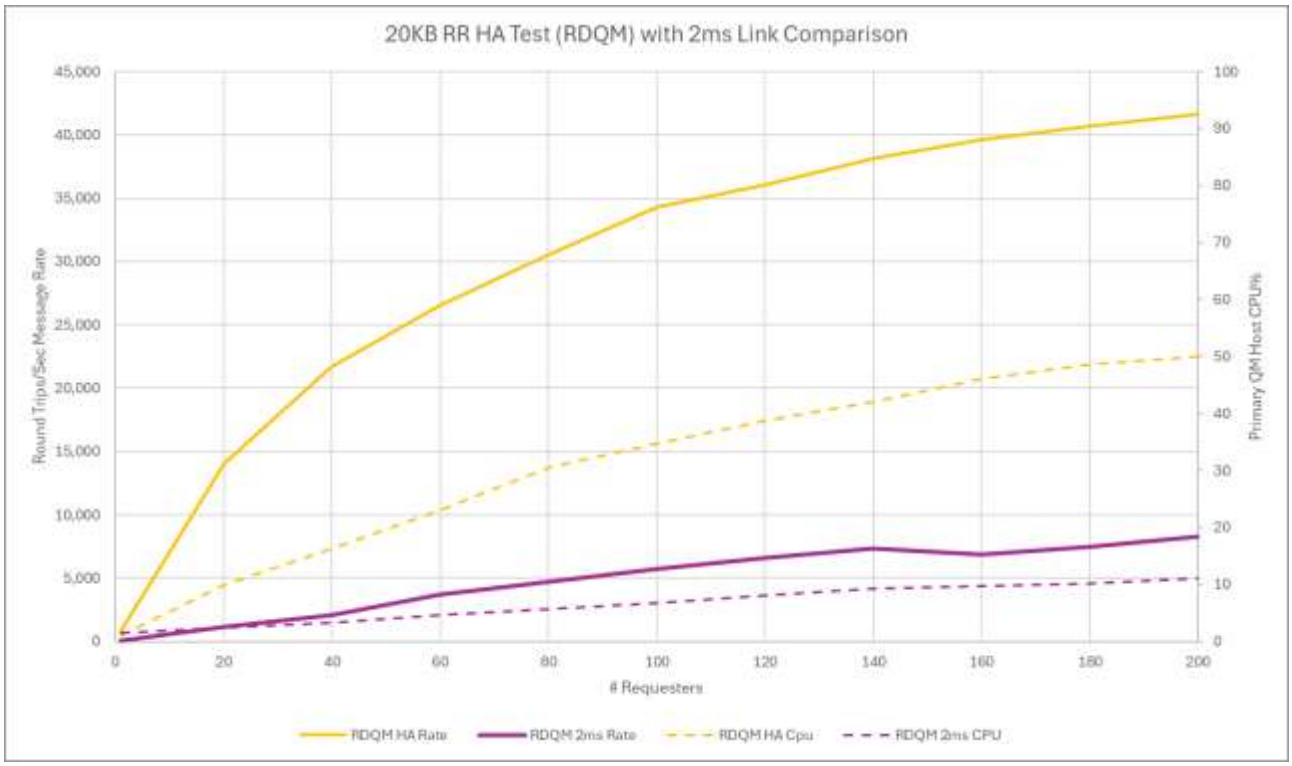


FIGURE 9 20KB RR RDQM(HA) TEST RESULTS WITH 2MS DELAY ON REPLICATION LINKS

As expected, in this messaging scenario, a 2ms delay on a fast, high bandwidth link reduced the throughput dramatically.

6 Differences in Replication Approaches of NHA and RDQM

NHA and RDQM take different approaches in ensuring data is replicated to the alternate instances of the queue managers in the HA group.

In either case data must be synchronously replicated to ensure integrity but RDQM replicates at the filesystem level (via DRBD) whilst NHA ensures data replication by only sending recovery log data to the replica queue managers.

Both NHA and RDQM will ensure that the replicas have a consistent view of all the persistent messages on the active instance, and no messages are lost or duplicated when switching between instances of a QM. If a persistent message has been committed on the active (primary) queue manager, then it will be available to the replica (secondary) queue manager if that is switched to. And the same applies when a persistent message is removed from that queue manager.

Table 1 shows the pertinent data replicated by RDQM and NHA.

	Queue files	**Recovery logs**
RDQM **(circular logging)**	Queue files replicated (including persistent and any non-persistent messages written to the queue file)	Recovery log replicated (containing committed persistent messages)
NHA **(replicated logging)**	No replication (queue files are dynamically built by the replicas based on the recovery logs)	Recovery log replicated (containing committed persistent messages and media images)

TABLE 1: NHA AND RDQM DATA REPLICATION BY QOS OF MESSAGE

The different approaches of RDQM and NHA mean there can be significant differences to the amount of data replicated in some circumstances:

- Non-persistent messages will be replicated by RDQM if they reach the queue file.
- Persistent messages will be replicated via the recovery log for both RDQM and NHA.
- Persistent messages will additionally be replicated by RDQM via the queue file (when they are written to the queue file due to the checkpointing process or the queue buffer being full).
- Persistent messages will additionally be replicated by NHA in media images. Messages remaining on a queue for a long time will need to be included in media images multiple times, to bring the media image recovery point forward in the recovery log.

Note: Despite some non-persistent messages being replicated by RDQM, only under very specific circumstances will non-persistent messages be available following a queue manager restart. Non-persistent messages *may* be recoverable if a queue is configured with NPMCLASS(HIGH) and the queue manager is shut down in a very controlled way, allowing it to persist and replicate the data. Obviously, the purpose of HA technologies is to provide automatic restart of queue managers following any shut down, controlled or not. Therefore, recovery of non-persistent messages should not be relied on for either NHA or RDQM.

6.1 Impact of Deep Queues on NHA and RDQM Replication

MQ is at its most efficient when messages are consumed promptly and data does not build up on the queues. There are circumstances where this may occur however, and this may impact performance when large amounts of data are replicated.

The performance impact of deep queues will depend on whether those queues are 'inactive' (i.e. a large amount of data in a queue not being accessed), or 'active' (messages being put and got from deep queues) and whether the messages are persistent or not.

6.1.1 'Inactive' Deep Queues

When a deep queue is not being used there will be no additional access to the underlying queue files after the queues have initially been populated. If the messages are persistent then they will be included in media images recorded in the recovery log by NHA however. In this use case we would expect to see no impact on an RDQM test (assuming initial replication of the queue files has completed) but some impact on NHA performance where the queue is filled with persistent messages.

6.1.1.1 NHA

NHA, with deep queues holding *persistent messages* will result in large media images being recorded in the recovery log. This can impact performance, especially on a heavily loaded system, running at or near to full capacity.

Figure 10 shows the impact of deep queues (large amount of 20K messages on additional queues) on the previously presented HA rates for 20K , persistent messaging.

The scenarios shown are:

Scenario	Description
NHA	Three QM NHA-HA topology with unsecured replication links.
NHA (+10GB-NP)	Three QM NHA-HA topology with additional 10GB of non-persistent messages on queues.
NHA (+10GB-P)	Three QM NHA-HA topology with additional 10GB of persistent messages on queues.
NHA (+1GB-P)	Three QM NHA-HA topology with additional 1GB of persistent messages on queues.

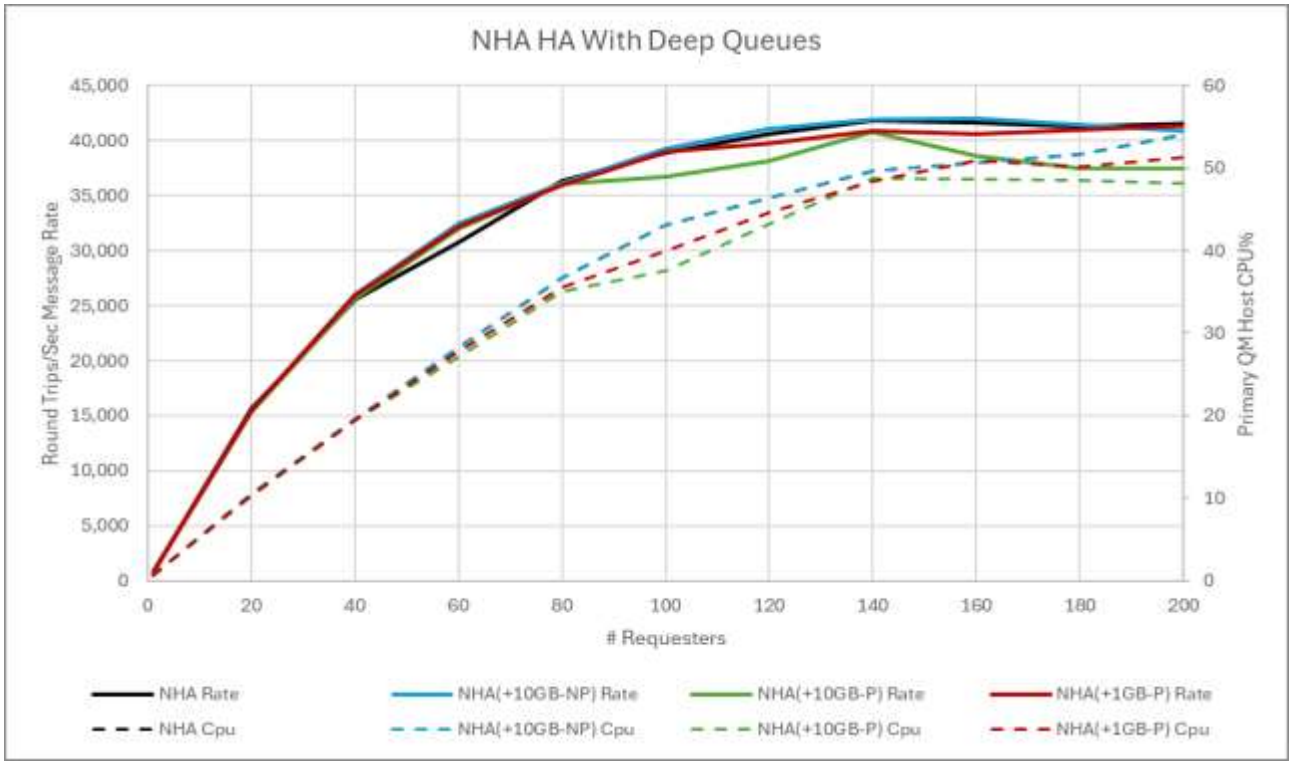


FIGURE 10 - IMPACT OF DEEP QUEUES ON NHA

Messaging rates were hardly affected by 10GB of non-persistent data as there is no additional data replication caused by this. 1GB of persistent data on the queues did not affect performance significantly either (despite the additional media image data that would be recorded during the run). When there is a larger amount of persistent data (10GB in this case) a small impact on performance became evident as the test reached it's limit. At lower numbers of requesters, the additional media image data did not noticeably affect performance.

Remember that these tests are with fast, 100Gb links between the HA queue managers in the data centre. The impact of deep queues will vary depending on the quality of your HA links, so performance testing your own environment is essential.

6.1.1.2 RDQM

With RDQM, there is no additional replication of data in media images once the data has been replicated on the queue files (messages will have spilled to the queue files in these tests as the queue buffers will have been exceeded by the deep queues).

Scenario	Description
RDQM	Three QM RDQM-HA topology with unsecured replication links.
RDQM (+100GB-P)	Three QM RDQM-HA topology with additional 100GB of persistent messages on queues.

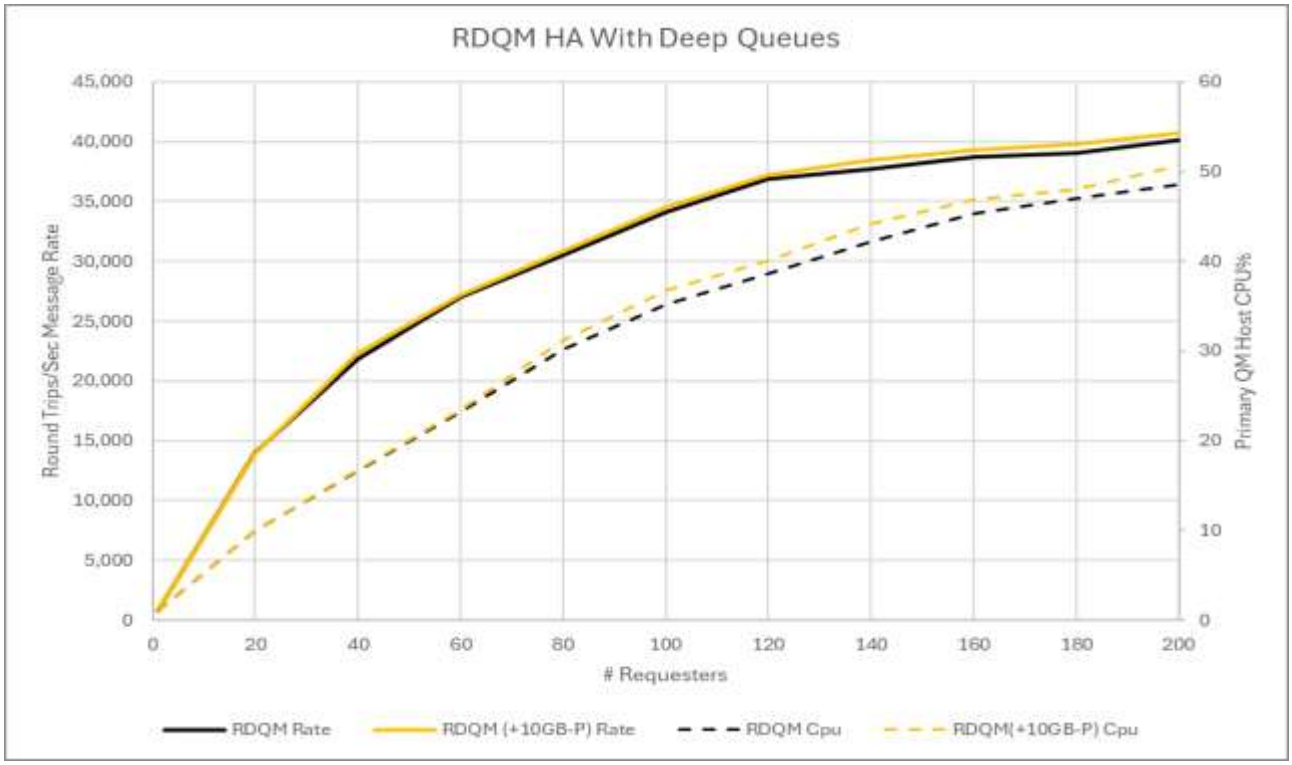


FIGURE 11 - IMPACT OF DEEP QUEUES ON RDQM

Even with 100GB of persistent data on the queues, no significant impact on performance was seen with RDQM. This is because once the data has been replicated in the recovery log and queue files, there is no ongoing replication required for messages sitting on the queues.

Note that as the messages are put onto the queues and spill the queue buffers because of the resulting depth of the queues, RDQM will replicate the data twice in the case of persistent messages but the test measured in Figure 11 was run after this.

With RDQM it's also possible to impact performance if a large amount of non-persistent data is put onto queues very quickly. Non-persistent messages can be added to queues as fast as the local (buffered) storage allows but the resultant queue files then need to be replicated. In one test where 100GB of non-persistent messages were dumped onto queues as fast as possible, a subsequent test slowed as the replication of the non-persistent data completed. This would be regarded as an edge-case however.

6.1.2 'Active' Deep Queues

This section highlights the effects on HA performance of deep queues being accessed by the queue manager, specifically deep queues holding non-persistent messages.

When deep queues are populated with a large amount of non-persistent data, they will be written to disk (the queue files). With RDQM this will be replicated, NHA will not be affected as no records are written to the recovery log.

If the deep queues are subsequently accessed by running a workload (putting an getting at a rate that sustains the depth of the queues) then there will be additional I/O to the disks hosting the queue files as messages are retrieved in a FIFO order, This activity on the queue files will be replicated by RDQM whereas NHA will not see the impact as only recovery log activity is replicated and these are non-persistent messages.

The deep queue scenario is run as follows:

1. 5000 200KB non-persistent messages are PUT onto 5 queues (1GB on each queue)
2. A background workload is run against the 5 deeply populated queues (rated at 500 PUTs and GETS per second).
3. The 20KB persistent requester/responder workload is run with the background workload from 2. Still active.

As a comparison, a similar background workload is run where the queues are ot previously populated with messages (so there is no activity on the underlying queue files as messages pass through the queue buffers).

6.1.2.1 NHA

A background non-persistent workload should exhibit minimal impact against an NHA deployment, given enough CPU capacity to run it.

Figure 12 shows the impact a background workload against deep and empty queues.

The scenarios shown are:

Scenario	Description
NHA	Three QM NHA-HA topology with unsecured replication links.
NHA (BG-DEEP)	Three QM NHA-HA topology with background NP workload against deep queues
NHA (BG)	Three QM NHA-HA topology with background NP workload against empty queues

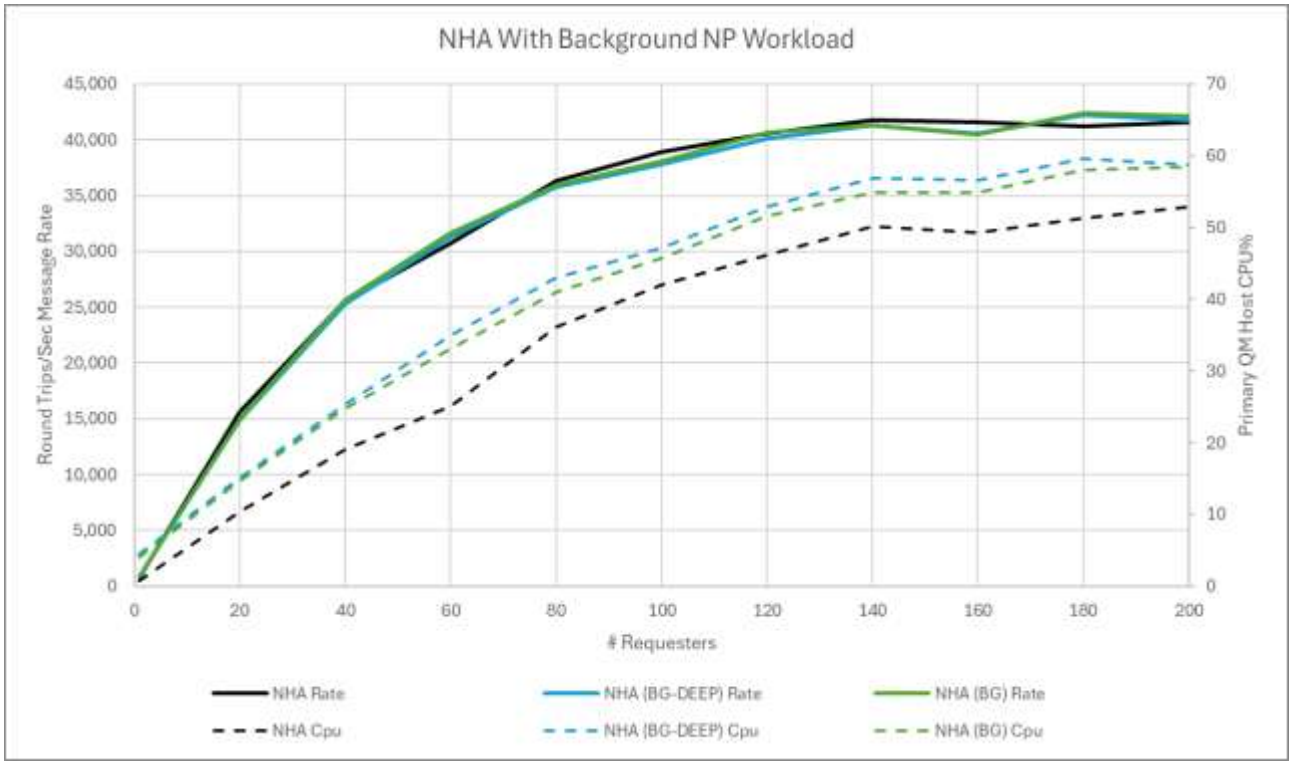


FIGURE 12 - IMPACT OF NON-PERSISTENT BACKGROUND WORKLOAD ON NHA

This demonstrates that if there is sufficient spare CPU capacity a background non-persistent workload does not impact NHA throughput. Whether the queues being accessed by the background workload are deep or effectively empty the only significant difference is the increased CPU consumption.

As with all scenarios presented in this report, your results may differ. If the background workload was running faster then CPU would become a bottleneck at some point and if the background workload was working with persistent messages then this would obviously impact the primary persistent workload due to the increased replication activity.

6.1.2.2 RDQM

With RDQM any activity to the files associated with the queue manger will be replicated to the replica QM filesystems. If a background workload is running against deep queues messages can be spilling the queue buffers and being written to disk. Activity on those queues will results in queue file changes being replicated by RDQM.

Figure 13shows the impact a background workload against deep and empty queues.

The scenarios shown are:

Scenario	Description
RDQM	Three QM RDQM-HA topology with unsecured replication links.
RDQM (BG-DEEP)	Three QM RDQM-HA topology with background NP workload against deep queues
RDQM (BG)	Three QM RDQM-HA topology with background NP workload against empty queues

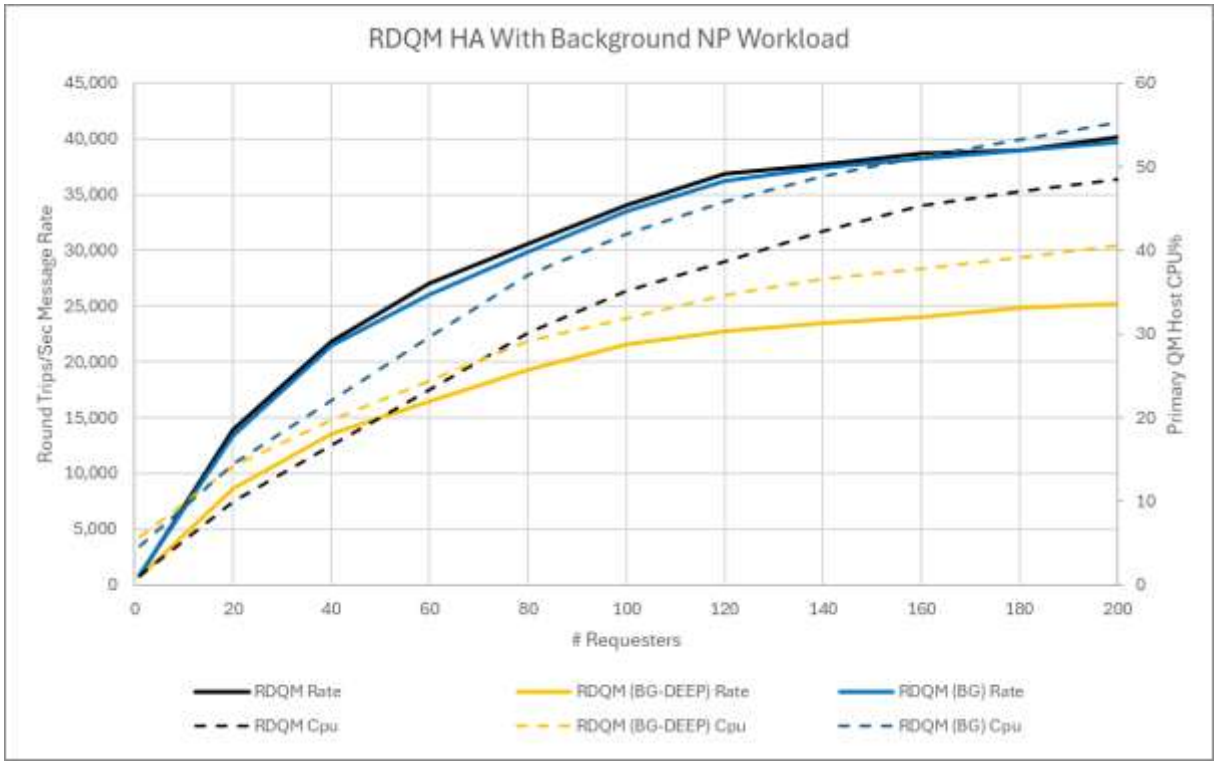


FIGURE 13 - IMPACT OF NON-PERSISTENT BACKGROUND WORKLOAD ON RDQM

This demonstrates that the background workload against empty queues only affects CPU (given sufficient spare CU capacity). When the background is run against deep queues however, the disk activity which must be replicated affects the throughput of the primary workload as it is additional to the recovery log replication.

A persistent background workload against deep queues would cause more replication, but that scenario would also affect NHA.

6.2 Steady State Comparison

All the results presented in the previous sections drive the hosts to saturation with a given number of clients. This gives an insight to how throughput limits are affected by concurrency (number of applications), encryption and the quality of the replication links.

In a production environment however, a workload is typically running at an expected rate within the capabilities of the environment. In this case, the interesting comparison is how much more processing power does it take to run with NHA or RDQM and what is the impact on response time.

For this comparison, the 20K persistent requester-responder scenario was run with 100 requesters, each running at a rate of 200 round-trips/sec (within the capabilities of the single server, NHA and RDQM deployments) The primary QM host CPU utilisation and response time of the clients were recorded

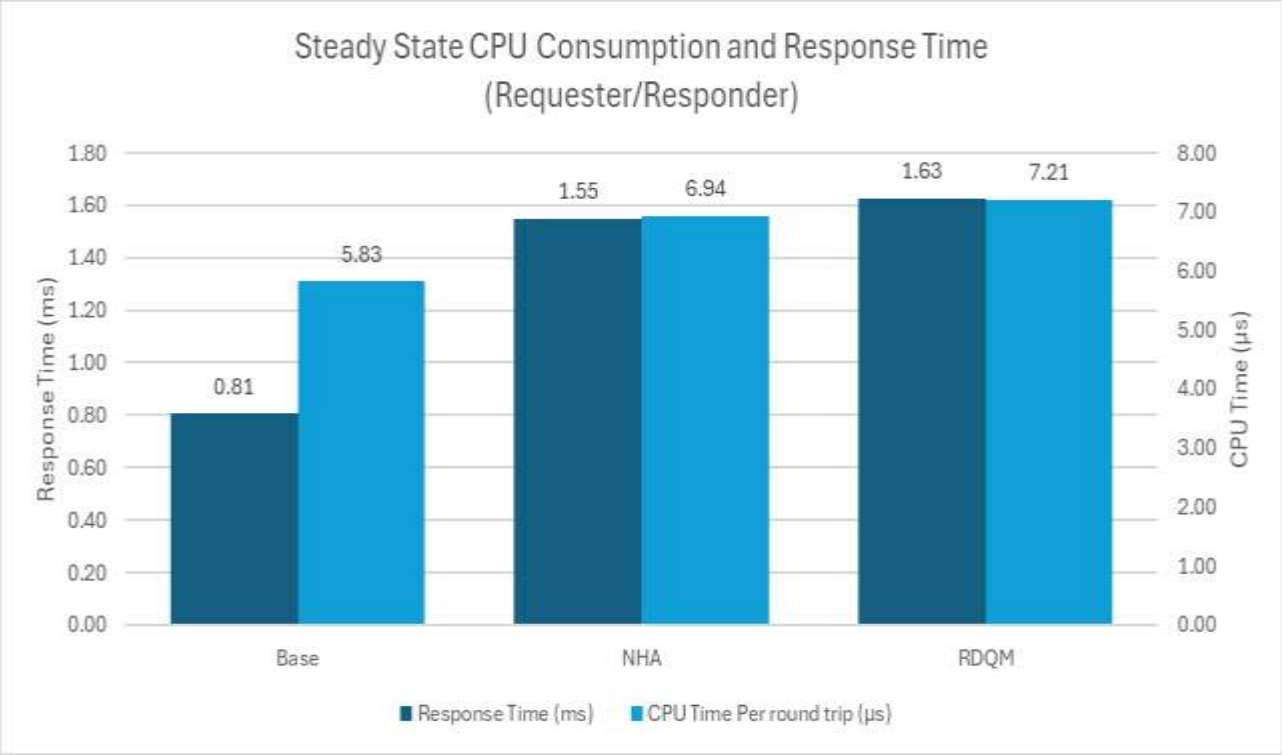


FIGURE 14 - STEADY STATE CPU CONSUMPTION AND RESPONSE TIMES

From the results of the steady state test (in this scenario at least) it can be seen that the CPU overhead of replication was between 18% and 26% whilst the impact on response time was more significant. This would be expected, given the requirement to synchronously replicate data over a network.

Remember that this is for a steady state of a well-behaved workload. A scenario where there were many persistent messages sitting on queue may impact NHA more, whilst lots of writes to queue files from deep non-persistent queues may impact RDQM detrimentally.

7 Conclusions

The HA replication functionality delivered in MQ by the NHA and RDQM options provide advanced capability to optimise critical business data availability in MQ, but with the requirement to synchronously offload data to secondary/replica instances of the queue manager there is a overhead which needs to be understood.

Generally, systems should not be running a workload that is constrained by its current environment, i.e. is utilising 100% of available resources. In that situation, then NHA or RDQM may add CPU cost or increase the response time whilst still maintaining the overall throughput (see section 6.2). However, data replication does add additional overheads, and may slow down an existing workload. In particular, performance tests that measure the maximum throughput achievable in a particular environment will see a significant impact from using NHA or RDQM as presented throughout this report.

The capabilities of your own environment are critical and should be understood, along with the typical usage of MQ as presented by your applications.

Evaluate your environment thoroughly:

- Replication links and filesystems hosting the MQ recovery logs are key resources.
- Dedicated links for replication are desirable, to minimise impact from other traffic and the capability of such links needs to be understood.
- Use network tools to evaluate the capability of your links.
- Measure the performance of the filesystem hosting your recovery logs. You can use the MQLDT tool for this (see Appendix B:)
    - This is particularly useful in the case of an RDQM environment, where using this tool to write to the filesystem of the primary QM will also test the underlying replication (assuming you are writing to a directory in the replicated volume).
    - For NHA you may use MQLDT to understand the performance of the underlying disk hosting the MQ recovery log, **but this will not include the additional replication**.
- Run performance tests which match your production use case a closely as possible. Running a test to fill a queue and then drain it for instance will introduce additional data to replicate in media images for NHA that may not be present in production skewing your view of performance.
    - The MQI test harness used in this report (CPH) is available to run as a first step. There is also a similar JMS tool (see Appendix B:)

- Record metrics from MQ (including amqsrua output) from when workloads are running as expected, to compare with data should you experience any slowdown.

Appendix A: Test Configurations

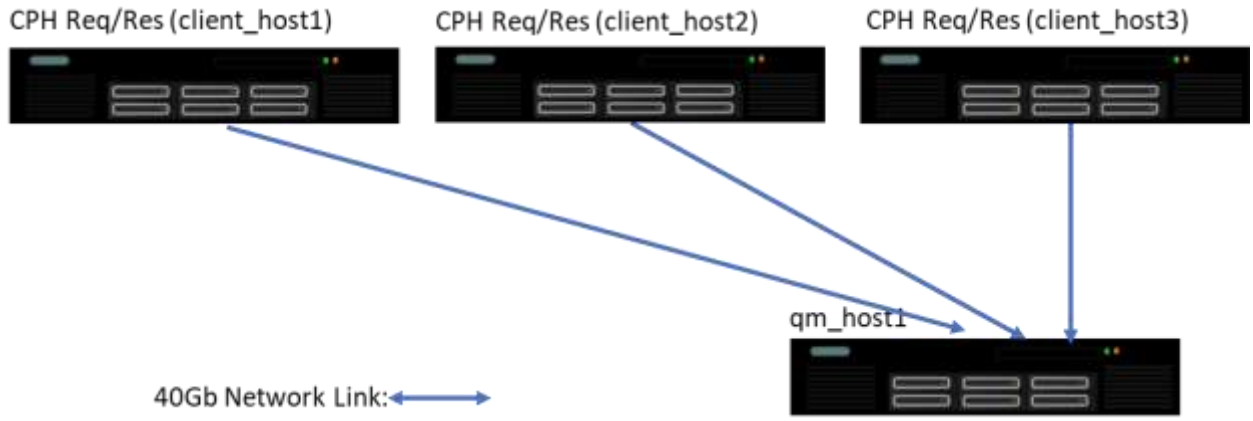
All tests in this report used the same set of machines for the clients and the same set for either:

1. The single QM server (baseline test)
2. The 'live' or 'active' set of HA hosts (for NHA or RDQM)

The topologies of the tests are detailed below with hardware and software specified in sections

A.1 Single Server Topology

The single server topology is used as a baseline to compare against the HA solutions of NHA and RDQM. The same three client machines drive a single instance queue manger on the same host that the HA active/primary QM uses in other tests.



For the hardware specifications of the hosts see section A.4

A.1.1 Software (all hosts)

Red Hat Enterprise Linux release 9.7 (Plow)

MQ-CPH MQI test driver (see Appendix B:)

IBM MQ V9.4.5

A.2 NHA Topology

The NHA tests used the following topology.

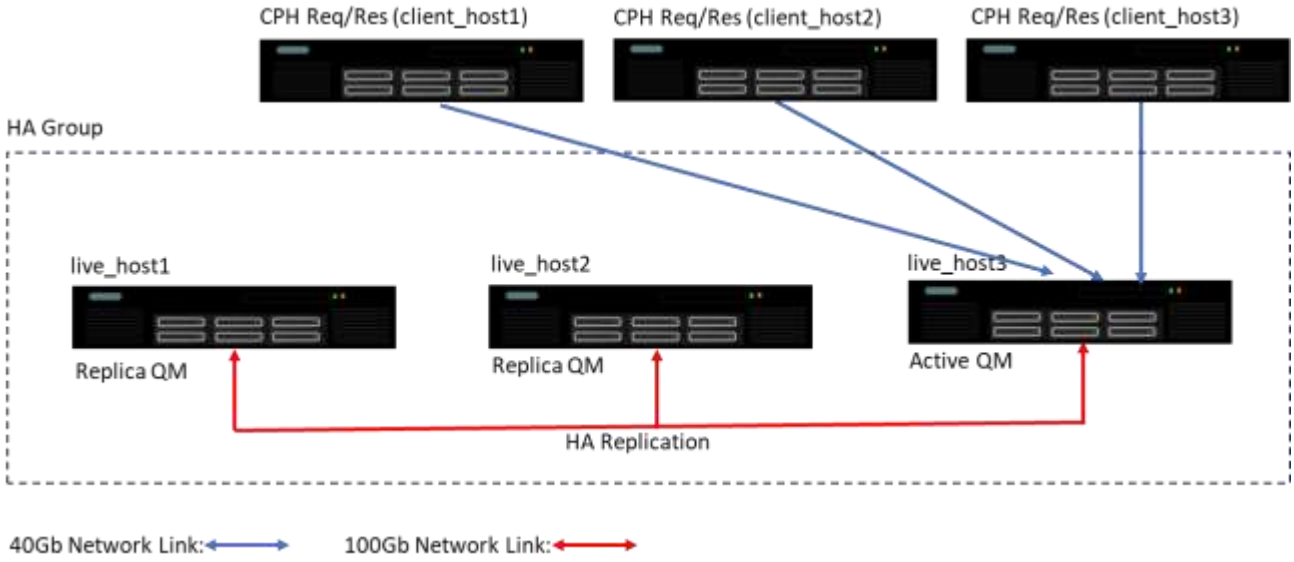


FIGURE 15 - NHA TEST TOPOLOGY

For the hardware specifications of the hosts see section A.4

A.2.1 Software (all hosts)

Red Hat Enterprise Linux release 9.7 (Plow)

MQ-CPH MQI test driver (see Appendix B:)

IBM MQ V9.4.5

A.3 RDQM 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 2 x 3.2TB NVMe SSDs (KCM61VUL3T20) in a RAID 0 array. Links between the applications and the RDQM nodes were 100Gb on the primary subnet, whilst the RDQM data replications links were 100Gb on a separate secondary subnet. Each RDQM node had a single Pacemaker address (HA_Primary), utilising the 100Gb link.

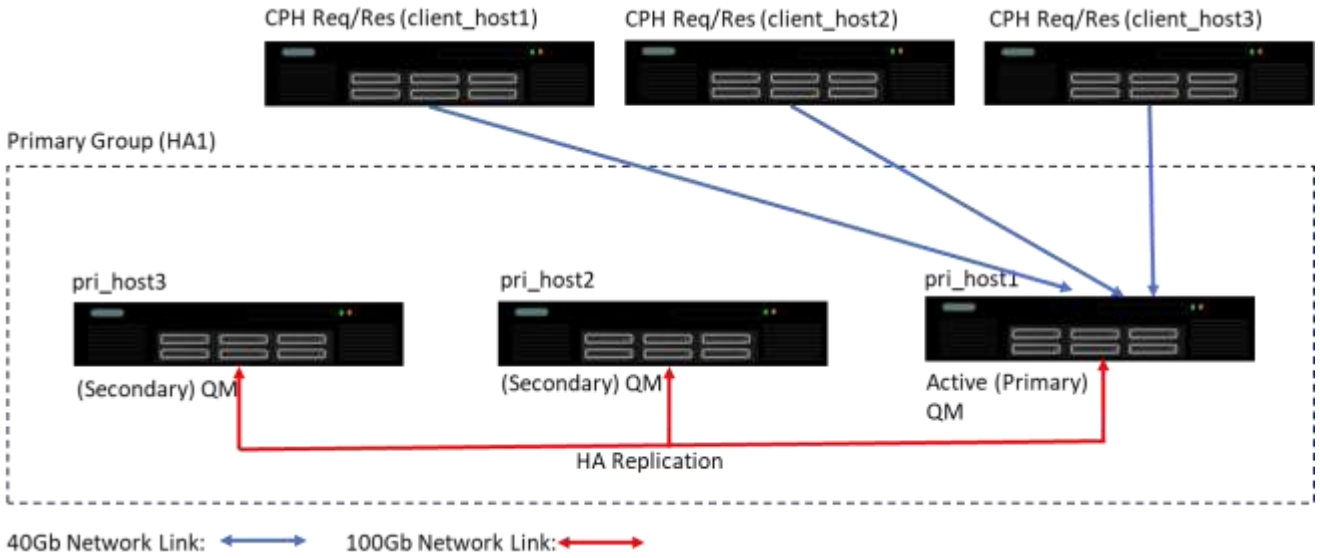


FIGURE 16 - RDQM TEST TOPOLOGY

For the hardware specifications of the hosts see section A.4

A.3.1 Software (all hosts)

Red Hat Enterprise Linux release 9.7 (Plow)

MQ-CPH MQI test driver (see Appendix B:)

IBM MQ V9.4.5

A.4 Hardware

client_host1, client_host2, client_host3:

- ThinkSystem SR630 - [7X02CTO1WW]
- 2 x 12 core CPUs.
  Core: Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
- 192GB RAM
- 1/10/40Gb Network links
- RAID-0 800GB SSD 2x370GB

Qm_host1 / live_host1, live_host2, live_host3 / pri_host1, pri_host2, pri_host3

- ThinkSystem SR630 V3 [7D73CTO1WW]
- 2 x 16 core CPUs
  Core: Intel(R) Xeon(R) Gold 6544Y CPU @3.60GHz
- 256GB RAM
- 2 x LENOVO 03GX685 - ThinkSystem 2.5in PM1655 800GB Mixed Use SAS 24Gb HS SSDs configured as RAID0 (Boot/OS partitions)
- 2 x 1.6TB NVMe SSD devices configured as a RAID0 array (/var/mqm filesystem on 1.5TB partition)
- 2 x 10Gb network links
- 2 x 100Gb network interfaces.

A.5 Tuning Parameters Set for Measurements in This Report

The tuning detailed below was set specifically for the tests being run for this performance report but in general follow the best practises.

A.5.1 Operating System

A good starting point is to run the IBM supplied program mqconfig. The following Linux parameters were set for measurements in this report.

/etc/sysctl.conf

```
fs.file-max = 19557658
net.ipv4.ip_local_port_range = 1024 65535
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.ipv4.tcp_rmem = 4096 87380 2147483647
net.ipv4.tcp_wmem = 4096 65536 2147483647
vm.max_map_count = 1966080
kernel.pid_max = 655360
kernel.msgmnb = 131072
kernel.msgmax = 131072
kernel.msgmni = 32768
kernel.shmmni = 8192
kernel.shmall = 18446744073692774399
kernel.shmmax = 18446744073692774399
kernel.sched_latency_ns = 2000000
kernel.sched_min_granularity_ns = 1000000
kernel.sched_wakeup_granularity_ns = 400000
vm.overcommit_memory = 0
```

/etc/security/limits.d/mqm.conf

```
@mqm soft nofile 1048576
@mqm hard nofile 1048576
@mqm soft nproc  1048576
@mqm hard nproc  1048576
```

A.5.2 IBM MQ

The following parameters are added or modified in the qm.ini files for the tests run in section 0 of this report:

```
Channels:
   MQIBindType=FASTPATH
   MaxActiveChannels=5000
   MaxChannels=5000
Log:
   LogBufferPages=4096
   LogFilePages=32764
   LogPrimaryFiles=32
   LogSecondaryFiles=2
   LogWriteIntegrity=TripleWrite
TuningParameters:
   DefaultPQBufferSize=10485760
   DefaultQBufferSize=10485760
   MinReusableLogExtents=750    ←NHA only
```

For NHA tests the LogType is defined as REPLICATED with LogManagement=Automatic

For RDQM test the LogType is defined as CIRCULAR.

All client channels were configured with SHARECNV(1), which is the recommendation for performance.

Appendix B: Resources

MQ Performance GitHub Site
https://ibm-messaging.github.io/mqperf/

IBM MQ Test Harnesses Launch Page (includes links to containerised versions of MQ-CPH & JMSPerfHarness) :
Test Harnesses.

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

Tutorial: MQ-CPH_Introduction.pdf

JMSPerfHarness (The IBM MQ Performance Harness for JMS)
https://github.com/ot4i/perf-harness

Tutorial: jmsperfharness_tutorial1.md

MQ Log Disk Tester (MQLDT)

https://github.com/ibm-messaging/mqldt