IBM MQ V9.4 (FP10) for AIX on Power Performance Report

Version 1.0 - July 2025

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, May 2025.

This edition applies to *IBM MQ V9.4.0.10* (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:

- Will be designing and implementing solutions using IBM MQ v9.4 for AIX on Power10.
- Want to understand the performance limits of IBM MQ v9.4 for AIX on Power10.
- Want to understand what actions may be taken to tune IBM MQ v9.4 for AIX on Power10.

The reader should have a general awareness of the AIX operating system and of IBM MQ 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, with regards to application design, MQ topology etc, is no longer included in the platform performance papers. A separate paper on general performance best practises has been made available here:

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

Contents

This report includes:

Release highlights with performance charts for IBM MQ V9.4 with Fix Pack 10 (V9.4.0.10), the latest available Fix Pack at time of testing.

- Performance measurements with figures and tables to present the performance capabilities of IBM MQ, across a range of message sizes, and including distributed queuing scenarios.

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

IBM MQ V9.4 is a long term service (LTS) release of MQ, which includes features made available in the V9.3.1, V9.3.2, V9.3.3, V9.3.4 & V9.3.5 continuous delivery (CD) releases.

All V9.4 tests in this report were run with Fix Pack 10 applied (i.e. IBM MQ V9.4.0.10)

Performance data presented in this report does not include release to release comparisons, but all tests run showed equal or better performance than the V9.3 release of IBM MQ.

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.

2 Release Highlights

Release highlights are listed in the MQ 9.4 documentation here:

https://www.ibm.com/docs/en/ibm-mq/9.4?topic=mq-whats-new-changed-in-940

2.1 Environment variables for tuning I/O operations that take too long

From IBM MQ 9.4.0, three new environment variables are added to increase or decrease the threshold at which a warning message is written to the queue manager log if a slow read/write time is detected. Fine tuning with these environment variables can help with diagnosing operating system or storage system issues and reduce the number of errors that are written to the log. For more information, see AMQ_IODELAY, AMQ_IODELAY_INMS and AMQ_IODELAY_FFST.

For example, if the following 2 environment variables are set, then warnings will be written to the queue manager error log, when a recovery log write takes over 7000µs (7ms).

```
export AMQ_IODELAY_INMS=YES
export AMQ_IODELAY=7000
```

Using these values for a test where the recovery log is hosted on nfs (on an x64 Linux server), we can introduce a latency of 10ms on the link from the Linux command line (the interface for the nfs link is ‘ens3f0’ in this case):

```
tc qdisc add dev ens3f0 root netem delay 10000us
```

MQ will report the long write time to the recovery log:

```
25/02/25 14:32:14 - Process(19858088.4) User(mqperf) Program(amqzmuc0)
                    Host(server1.hursley.ibm.com) Installation(Installation1)
                    VRMF(9.4.0.10) QMgr(PERF0)
                    Time(2025-02-25T14:32:14.804Z)
                    ArithInsert1(4096)
                    CommentInsert1(W)
                    CommentInsert2(7000)
                    CommentInsert3(10465)

AMQ6729W: Log I/O operation (W) exceeded threshold (7000 microseconds).

EXPLANATION:
A Read (R) or Write (W) of 4096 bytes took longer than expected to complete.
This might indicate a problem with your O/S or storage system. If this occurs
frequently, queue manager performance is likely to be severely impacted.
ACTION:
Investigate cause of long I/O times in your storage provision.  If these delays
are expected in your environment, the warning threshold can be increased by
modifying the AMQ_IODELAY environment variable.
----- amqhose0.c : 106 ------------------------------------------------------
```

Note the 2nd and 3rd ‘CommentInsert’ fields in the message (highlighted in red), which show the current value of AMQ_IODELAY in micro-seconds and the length of time the write operation took that triggered this message.

Setting AMQ_IODELAY to a sensible value (determined by expected peak latency during a ‘normal’ day) enables you determine when the recovery log filesystem may have issues.

2.2 LZ4 Compression Options

With MQ V9.4, new LZ4 compression algorithms are available, see the IBM Integration Community article here for the general details.

For AIX on Power9 and up, an enhanced version of the zlib compression library (zLinbNX) that supports hardware-accelerated data compression and decompression by using co-processors can be used. [a]

Whilst the zlibNX library can deliver a performance boost, it was not the best option here. The RR-CC tests scale by adding in ever increasing numbers of client threads, all of which are utilising compression and can contend for use of the co-processor. ZLIBFAST numbers below are not utilising the zlibNX library.

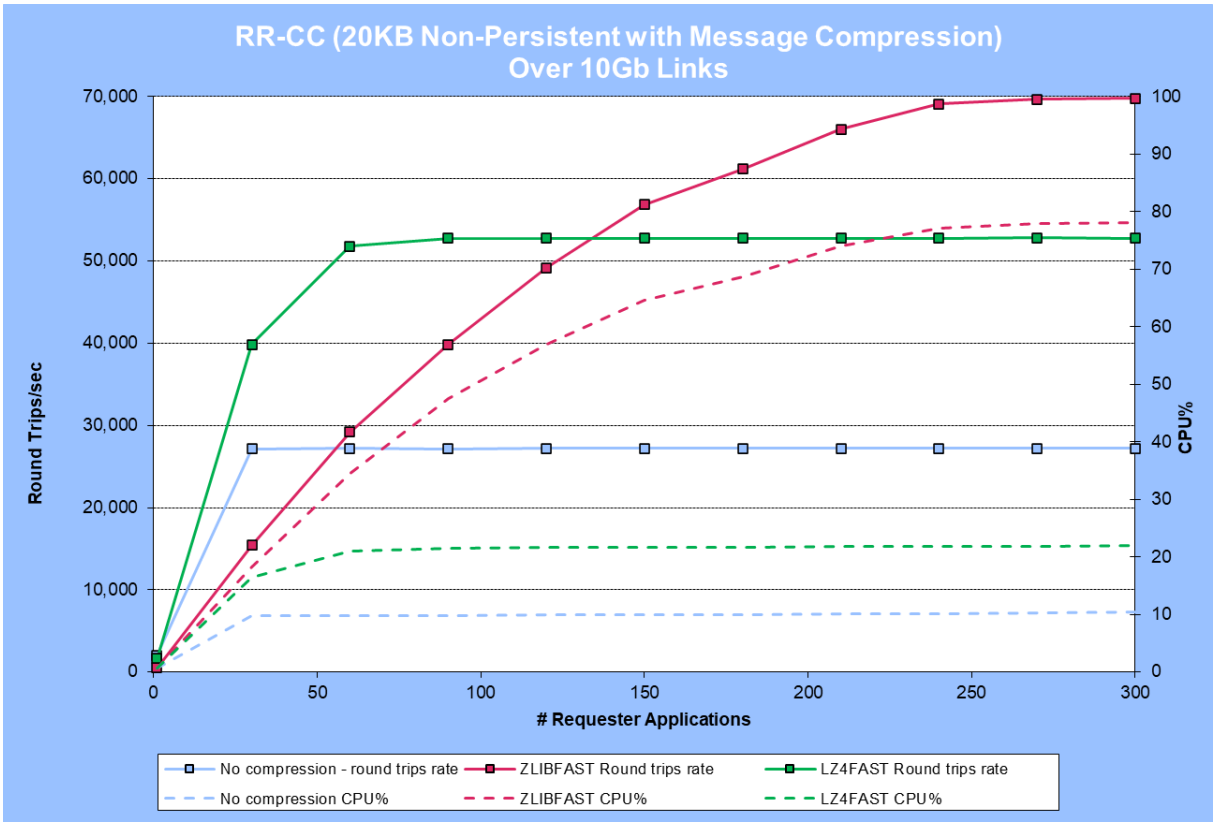

FIGURE 1- EFFECT OF MESSAGE COMPRESSION OVER NARROW BANDWIDTH NETWORK CONNECTION (10GB ETHERNET)

Results presented throughout this report were run against hosts in the same datacentre and connected via 100Gb network links. In such an environment, setting message compression is unlikely to increase the throughput, but for smaller bandwidth links, the effects can be significant.

[a] To enable a message channel agent (MCA) to use the zlibNX library, set the environment variable AMQ_USE_ZLIBNX.

Figure 1 above show the effects of using the ZLIBFAST or LZ4FAST compression algorithms on the SVRCONN channels used by the requester and responder applications for 20K messages. The FAST options were found to be more beneficial in this environment for both algorithms (rather than using ZLIBHIGH or LZ4HIGH). Note that the new LZ4 algorithm available in MQ V9.4 was a lot faster for smaller numbers of clients, consuming much less CPU and enabling a higher message rate to be achieved through the bottleneck of the 10Gb network link. For a larger number of clients (i.e. greater concurrently), the existing ZLIBHIGH algorithm achieved the higher throughput, but at greater cost in CPU terms. You should determine which compression algorithm suits your workload through testing.

The benefit of using compression will depend on several factors including:

- Size of message
- Quality of network link (bandwidth and latency).
- Compressibility of the message data.
- Available CPU resource (for both the queue manager host and the hosts where the applications are running).

For the 10Gb linked environment for the 20K results above, varying degrees of benefit were recorded when using compression.

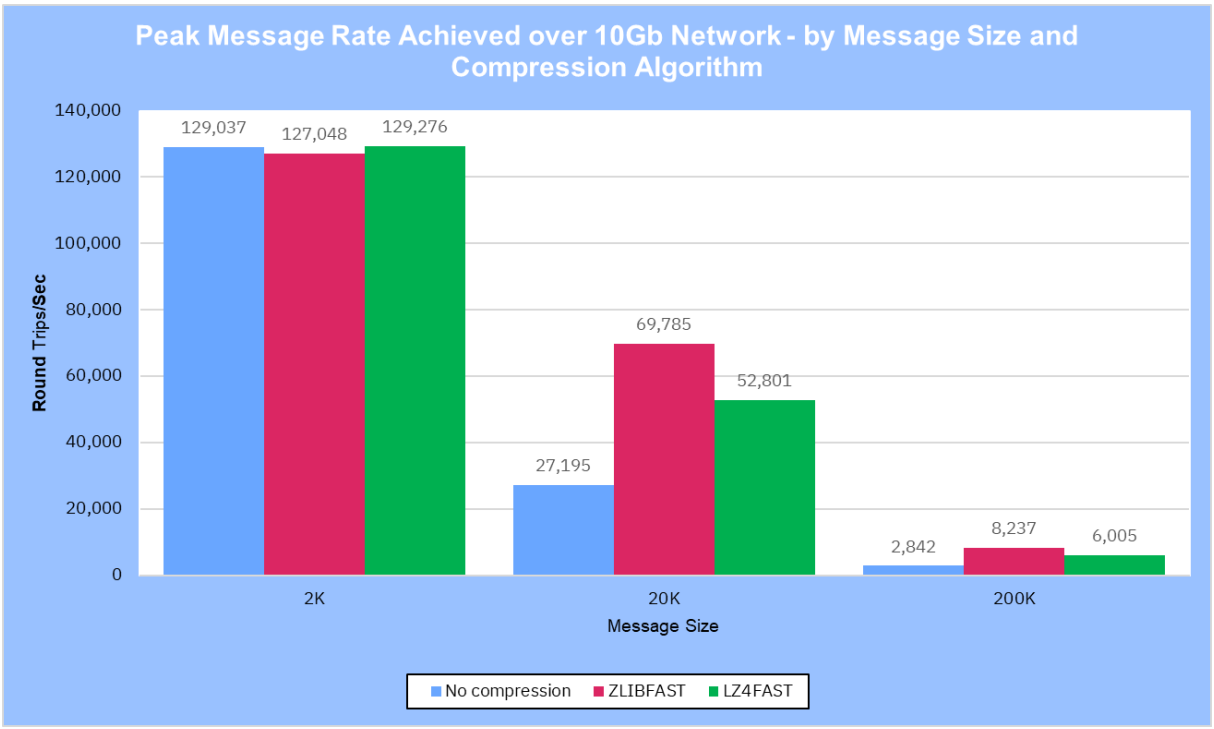

FIGURE 2- PEAK MESSAGE RATES BY MESSAGE SIZE AND COMPRESSION ALGORITHM.

Figure 2, shows the peak rate achieved for 2K, 20K and 200K messages, without compression vs compression (ZLIBFAST or LZ4FAST). For 2KB messages there was no significant improvement utilising compression.

For larger messages (20KB and 200KB), utilising compression gave significant throughput benefits. Whilst the ZLIBHIGH gave the best throughput, greater numbers of clients were required and the CPU cost was higher. In the 200KB case 90 requester applications were

needed before ZLIBHIGH achieved a greater throughout than LZ4FAST, at which point it was using 3x the CPU of LZ4FAST.

As stated above, compressibility of the message will be a factor. For these tests the message data was comprised of JSON text. Binary data will not benefit as much (or at all) as this is often of an uncompressible nature (e.g. already a compressed format). Other messages may be more compressible and exhibit a greater benefit.

The table below shows the compression rates achieved (as reported by the COMPRATE value in the channel status). Although the HIGH variants of the algorithms compressed these JSON messages a little more, they did not give as much benefit as the FAST variants due to the additional time taken for the more aggressive compression.

Msg Size	Compression Rate			
	ZLIBFAST	LZ4FAST	ZLIBHIGH	LZ4HIGH
2KB	34%	17%	35%	19%
20KB	56%	43%	58%	47%
200KB	59%	46%	61%	52%

TABLE 1 - MESSAGE COMPRESSION RATES

The new LZ4 compression algorithm performed significantly better than the existing ZLIB algorithm for these tests at lower numbers of clients. At higher numbers of clients, the ZLIB algorithms may achieve more throughput, but at a significantly higher CPU cost. If your environment is constrained by the network between the clients and the queue manager, it is worth testing with compression to measure the potential benefit.

Note that compression algorithms can also be set on channels between queue managers, so consider using them for intra-queue manager traffic where the network bandwidth is a limiting factor.

2.2.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1, Client 1, Client 1 (see section A.1).

3 Base MQ Performance Workloads

Table 2 (below) lists the workloads used in the generation of performance data for base MQ (that is standard messaging function) in this report. All workloads are requester/responder (RR) scenarios which are 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. They typically run 'unrated' (no think time between getting a reply and putting the next message on the request queue).

Workload	Description
RR-DQ-BB	Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders.
RR-CC	Client mode requesters, and responders on separate, unique hosts

TABLE 2 - WORKLOAD TYPES

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

RR-CB & RR-DQ-BB are described in the following section. The remaining two workloads differ only in the location of the MQ applications, which is made clear in the results presented in this report.

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.

3.1 RR-CC Workload

(Client mode requesters with client mode responders.)

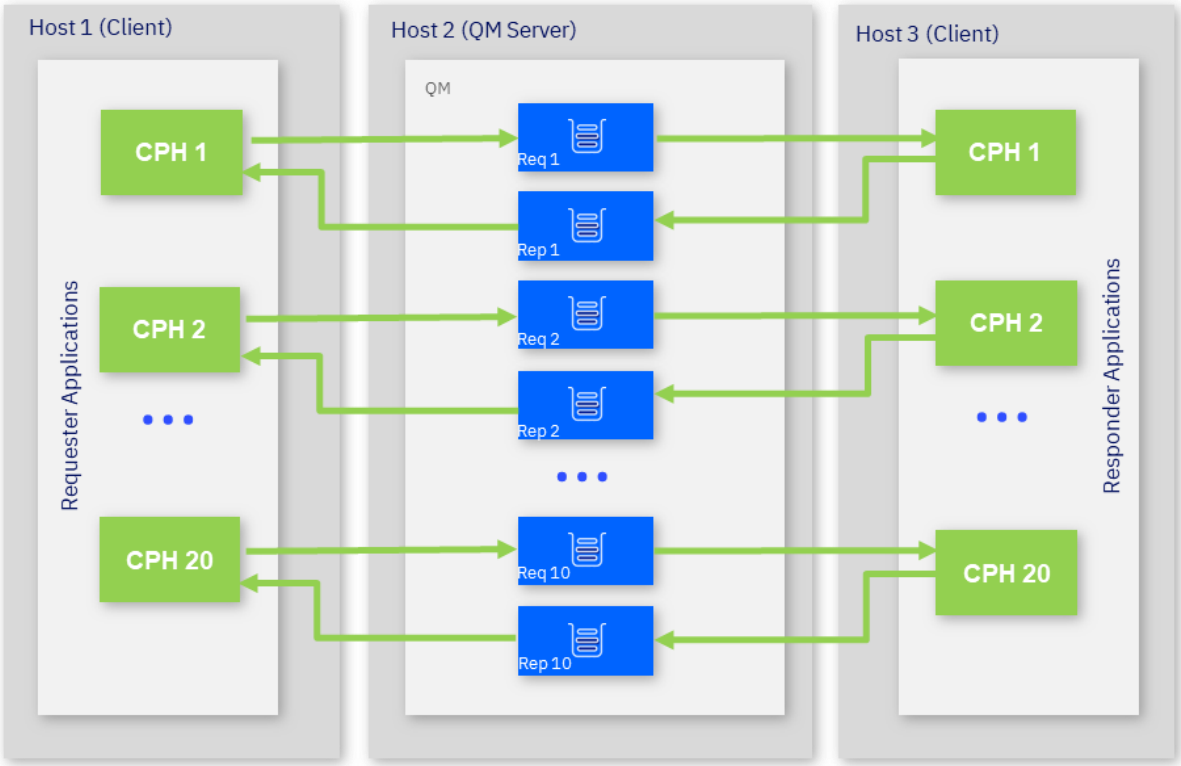

FIGURE 3 - RR-CC TOPOLOGY

Figure 3 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.

Note that for a lot of the tests in this AIX report, the requesters and responders were hosted on separate x64 Linux machines, to illustrate how far the AIX server would scale (these tests would have otherwise peaked when the single AIX client machine reached 100% CPU). Each set of results presented in this report is accompanied by the hardware used in the test.

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.

3.2 RR-DQ-BB Workload

(Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders).

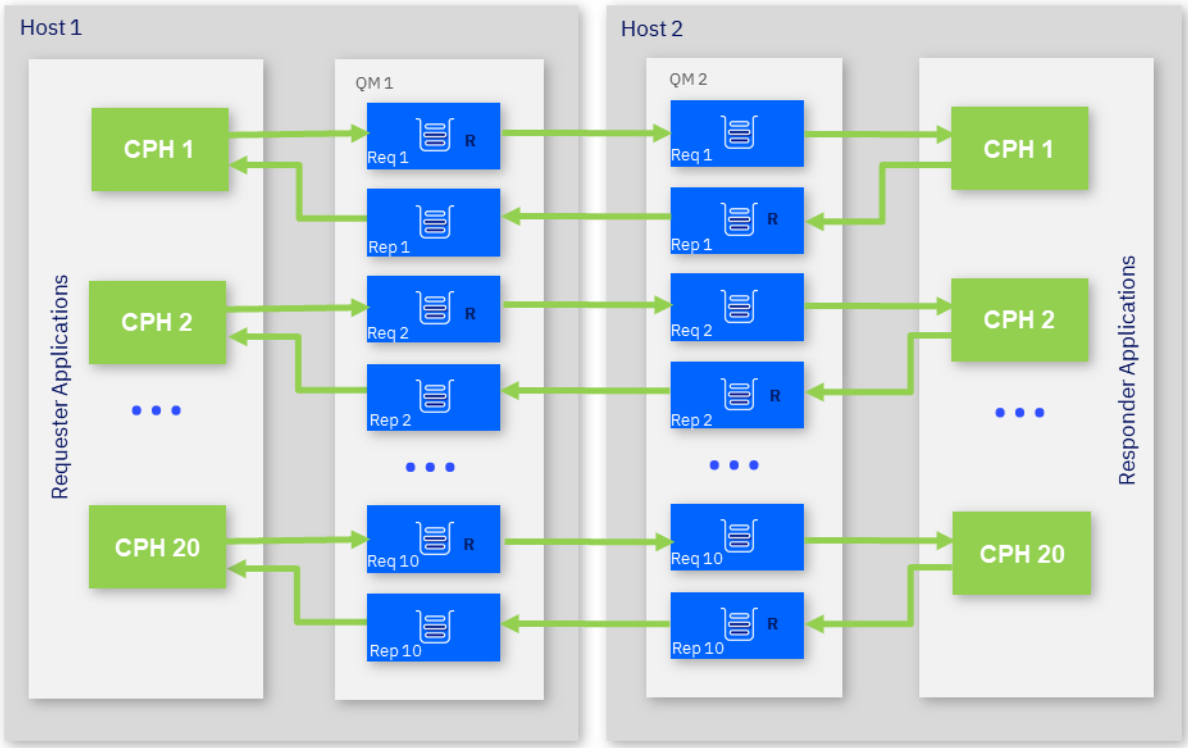

FIGURE 4 - REQUESTER-RESPONDER WITH REMOTE QUEUE MANAGER (REMOTE RESPONDERS).

This is a distributed queuing version of the requester-responder topology detailed in section 3.2. All *MQPUT*s are to remote queues (marked with 'R' in the diagram above), so messages are now transported across server channels to the queue manager where the queue is hosted. Note that remote queues are distributed across multiple pairs of sender/receiver channels in the tests below, but a single pair or channels may be adequate in your installation.

Network Flows:

As for the RR-CC topology, each round trip will comprise of 2 inbound messages to the server and 2 outbound messages from the server, but as the clients are local to the queue manager these do not utilise network bandwidth. For each round trip there will be a single outbound and inbound message between the queue managers across the network. So if the message size is 2048 bytes there will be 1 x (2048 + metadata) inbound to the MQ server and 1 x (2048 + metadata) outbound from the server, where metadata is the non-message payload data, comprising of the MQ and transport headers. This scenario therefore uses half the network bandwidth of RR-CC for a given message rate.

4 Non-Persistent Performance Test Results

Full performance test results are detailed below. The test results are presented by broad categories with an illustrative plot in each section followed by the peak throughput achieved for the remaining tests in that category (the remaining tests are typically for different message sizes).

4.1 RR-CC Workload

The following chart illustrates the performance of 2KB non-persistent messaging with various numbers of requester clients.

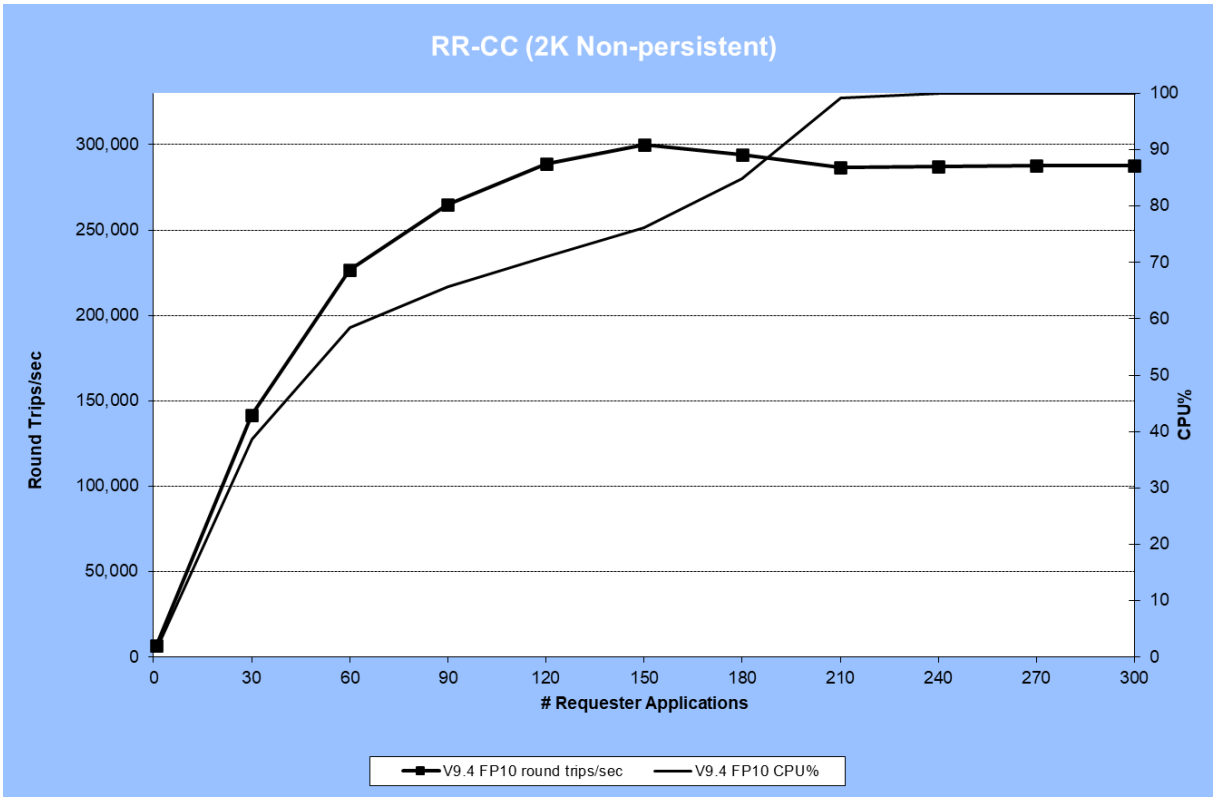

FIGURE 5 - PERFORMANCE RESULTS FOR RR-CC (2KB NON-PERSISTENT)

The test peaked 300,000 round trips/sec, with the MQ server CPU at around 76% CPU utilisation.

Peak round trip rates for all message sizes tested can be seen in the table below.

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-CC (2K Non-persistent)	300,110	76.13	150
RR-CC (20K Non-persistent)	185,016	89.68	350
RR-CC (200K Non-persistent)	21,367	45.38	60
RR-CC (2MB Non-persistent)	1,547	24.8	50

Round trips/sec

TABLE 3 - PEAK RATES FOR WORKLOAD RR-CC (NON-PERSISTENT)

4.1.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux) - see section A.1.

4.2 RR-DQ-BB Workload

(Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders).

The distributed queuing scenarios use workload type RR-DQ-BB (see section 3.2) where locally bound requesters put messages onto a remote queue.

The throughput will be sensitive to network tuning and server channel setup amongst other things. All the tests in this section utilise multiple send/receive channels. This particularly helps with smaller, non-persistent messages when the network is under-utilised.

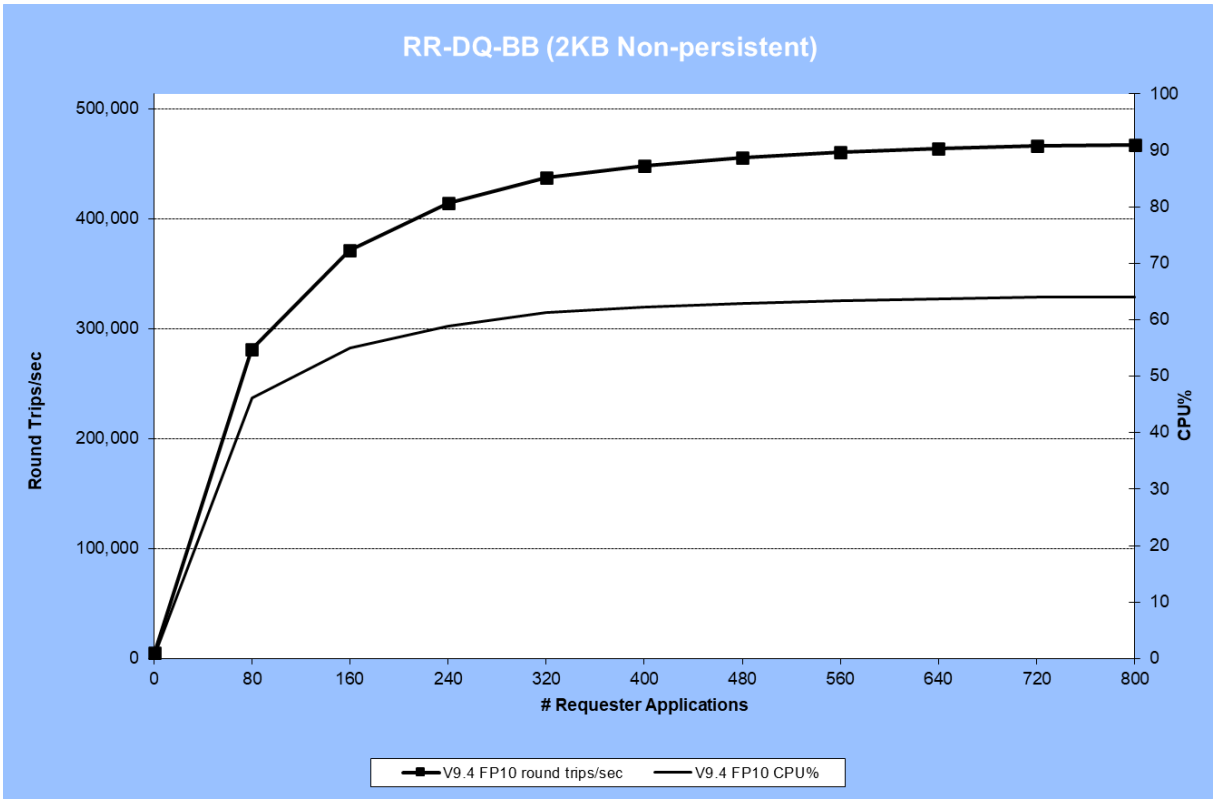

FIGURE 6 - PERFORMANCE RESULTS FOR RR-DQ-BB (2KB NON-PERSISTENT)

The distributed queuing test exhibits good scaling to around 64% of the MQ server, suggesting that multiple QMs would be needed to saturate the network bandwidth for a server of this size.

Peak round trip rates for all message sizes tested can be seen in the table below. Note that these rates are higher than the RR-CC test in the previous section as the overall network traffic is lower per message (see the notes on network traffic in sections 3.1 and 3.2)

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-DQ-BB (2KB Non-persistent)	467,291	64.03	800
RR-DQ-BB (20KB Non-persistent)	237,820	57.07	180
RR-DQ-BB (200KB Non-persistent)	38,036	41.3	60
RR-DQ-BB (2MB Non-persistent)	2,848	25.95	30

Round trips/sec

TABLE 4 – FULL RESULTS FOR WORKLOAD RR-DQ-BB (NON-PERSISTENT)

4.2.1 Test setup

Workload type: RR-DQ-BB (see section 3.2).

Hardware: Server 1, Client 1 (see section A.1).

4.3 RR-CC JMS Workload

This test application is JMSPerfharness, which is run unrated (i.e. each requester sends a new message as soon as it receives the reply to the previous one). The JMS test is run with both requesters and responders in client mode on remote hosts as JMSPerfharness is a relatively resource hungry application, utilising multiple JVMs to scale up the JMS connections. JMS client applications are running on 2 x64 Linux hosts (see test configuration below) to maximise the throughput through the AIX QM server.

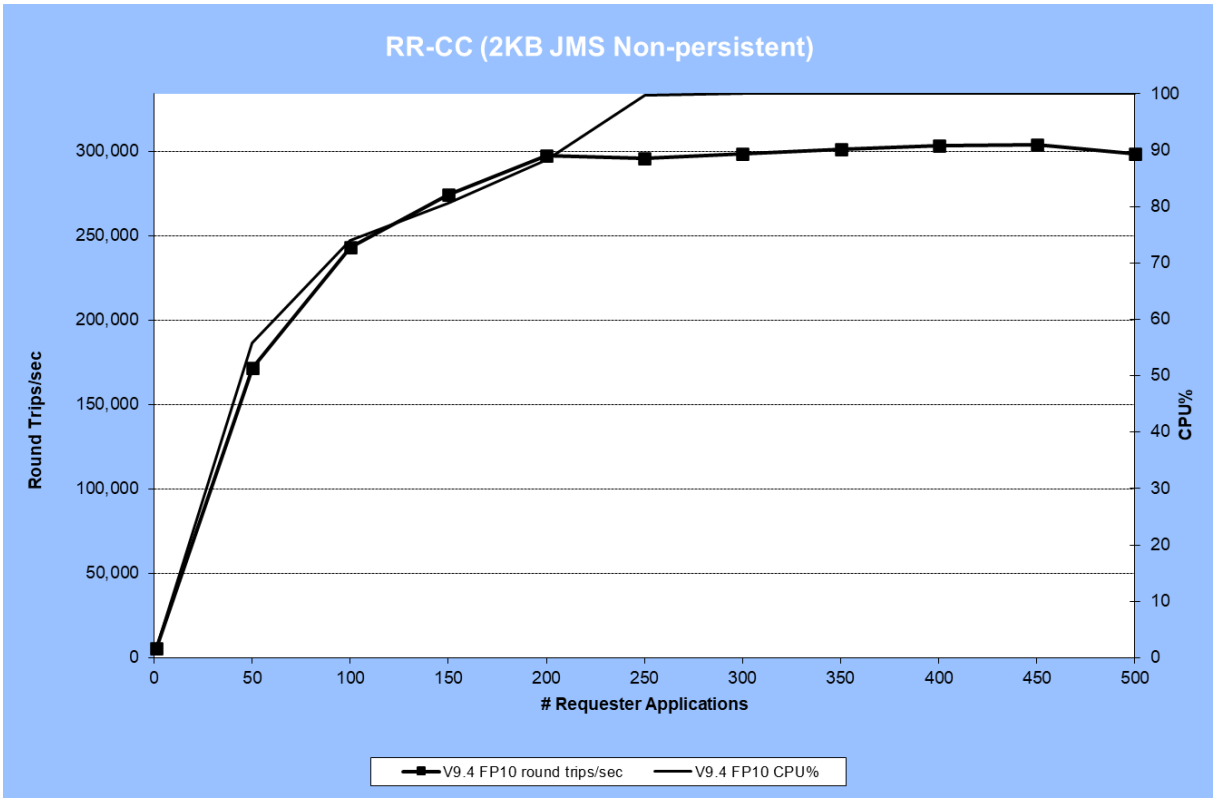

FIGURE 7 - PERFORMANCE RESULTS FOR RR-CC (2KB JMS NON-PERSISTENT)

This workload peaks at a similar throughput (about 300,000 round trips/sec) as the non-JMS workload, but required more clients and to reach that point and consumed more CPU.

Peak round trip rates for all message sizes tested can be seen in the table below.

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-CC (2KB JMS Non-persistent)	303,682	99.98	450
RR-CC (20KB JMS Non-persistent)	162,905	82.03	360
RR-CC (200KB JMS Non-persistent)	17,926	50.38	210
RR-CC (2MB JMS Non-persistent)	1,527	46.9	200

Round trips/sec

TABLE 5 - PEAK RATES FOR JMS (NON-PERSISTENT)

4.3.1 Test setup

Workload type: RR-CC (see section 3.1).

Message protocol: JMS

Hardware: Server 1, Client 2 (x64 Linux), Client 3 (x64 Linux) (see section A.1).

5 Persistent Performance Test Results

The performance of persistent messaging is largely dictated by the capabilities of the underlying filesystem hosting the queue files, and more critically, the MQ recovery log files. Writes to the recovery log need to be synchronous to ensure transactional integrity, but IBM MQ is designed to maximise throughput, by aggregating writes where possible. Aggregation of log writes is dependent on a concurrent workload (i.e. multiple applications connected and committing data to the queue manager concurrently, such that the MQ logger component can aggregate data into larger, more efficient file writes and mitigate the higher latency of some file systems).

The performance of persistent messaging is therefore dependant on the machine hosting MQ, the degree of concurrency, *and* the I/O infrastructure. Some comparisons are shown below between non-persistent and persistent messaging for local storage and then results for V9.4 FP10 in a separate environment (AIX with SSD, SAN or NFS filesystems) are shown to demonstrate the impact of recovery log location.

5.1 RR-CC Workload

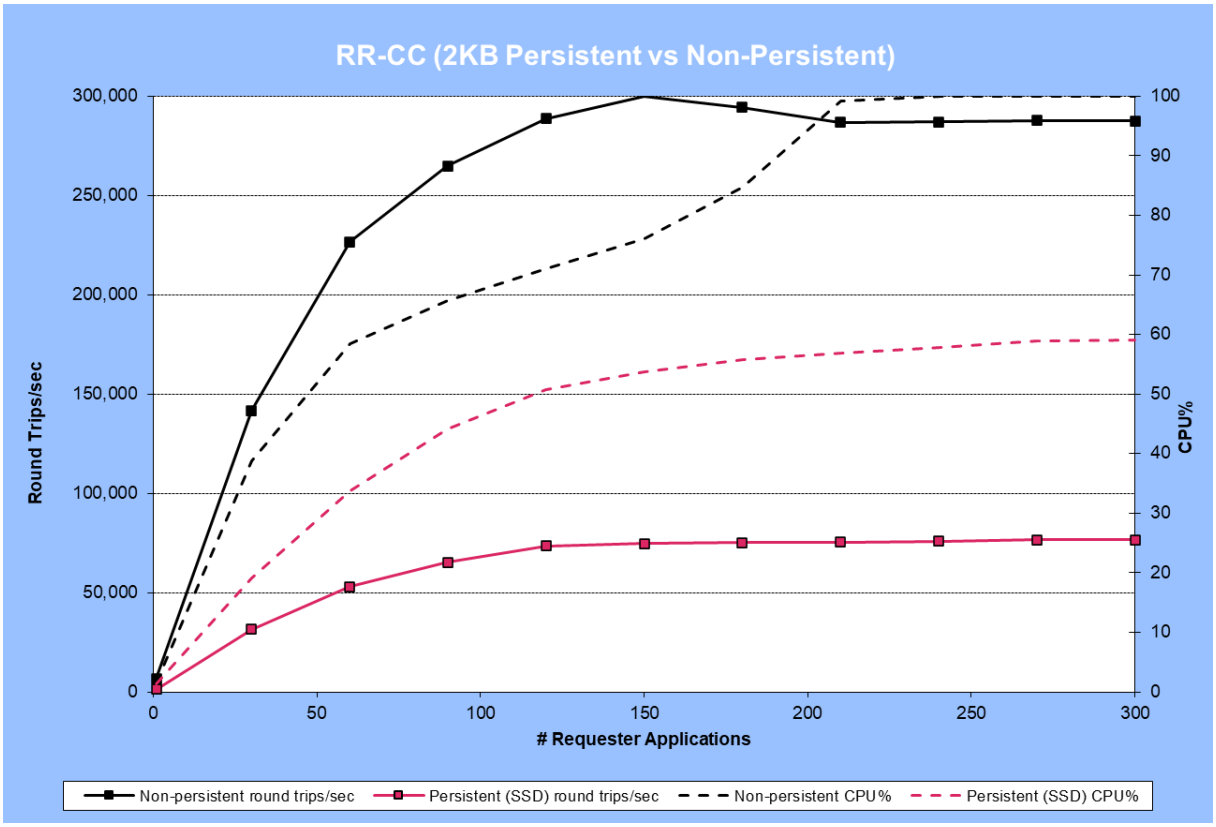

FIGURE 8 - PERFORMANCE RESULTS FOR RR-CC (2KB NON-PERSISTENT VS PERSISTENT)

Figure 8 shows results from running the RR-CC workload with 2KB non-persistent and persistent messages, on the same server used for the non-persistent scenarios in the previous sections.

The non-persistent workload reaches an optimal value at 150 requesters then tailing off as the CPU approaches 100% utilisation. Adding more requesters degrades performance, increasing context switching on an already saturated server.

Note that for smaller message sizes (as for 2KB, above), higher rates of throughput in persistent scenarios are attained when there is a greater deal of concurrency (i.e. requester applications) as this enables the logger to perform much larger writes (as described above).

In these tests, the machines are connected via 100Gb links in the same data centre. With network links that are higher latency or lower bandwidth, the difference between non-persistent and persistent throughput will be less, as the network becomes a significant part of the bottleneck.

Peak round trip rates for all message sizes tested, for persistent & non-persistent scenarios can be seen in Table 6 & Table 7 below.

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-CC (2K Non-persistent)	300,110	76.13	150
RR-CC (20K Non-persistent)	185,016	89.68	350
RR-CC (200K Non-persistent)	21,367	45.38	60
RR-CC (2MB Non-persistent)	1,547	24.8	50

***ROUND TRIPS/ SEC**

TABLE 6 - PEAK RATES FOR WORKLOAD RR-CC (NON-PERSISTENT)

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-CC (2KB Persistent - SSD)	76,686	59.13	300
RR-CC (20KB Persistent - SSD)	55,088	49.3	270
RR-CC (200KB Persistent - SSD)	6,871	16.68	90
RR-CC (2MB Persistent - SSD)	622	11.65	25

***ROUND TRIPS/ SEC**

TABLE 7 - PEAK RATES FOR WORKLOAD RR-CC (PERSISTENT)

The non-persistent numbers are for comparison with persistent messaging, to illustrate what the impact of logging can be.

The recovery log I/O is the limiting factor for the persistent workloads here, as expected. As the message size goes up, the time spent on the recovery log write becomes a larger factor, so although the bytes per sec is more, the overall CPU utilisation is lower. The level of concurrency needed to reach the limitations of the filesystem also drops as the message size increases.

5.1.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux) - see section A.1.

5.2 Impact of Different File Systems on Persistent Messaging Performance

If possible, you should assess the performance of a new application, with non-persistent messaging first. If the target rate of messaging is met, then calculate the required bandwidth of the filesystem hosting the recovery logs.

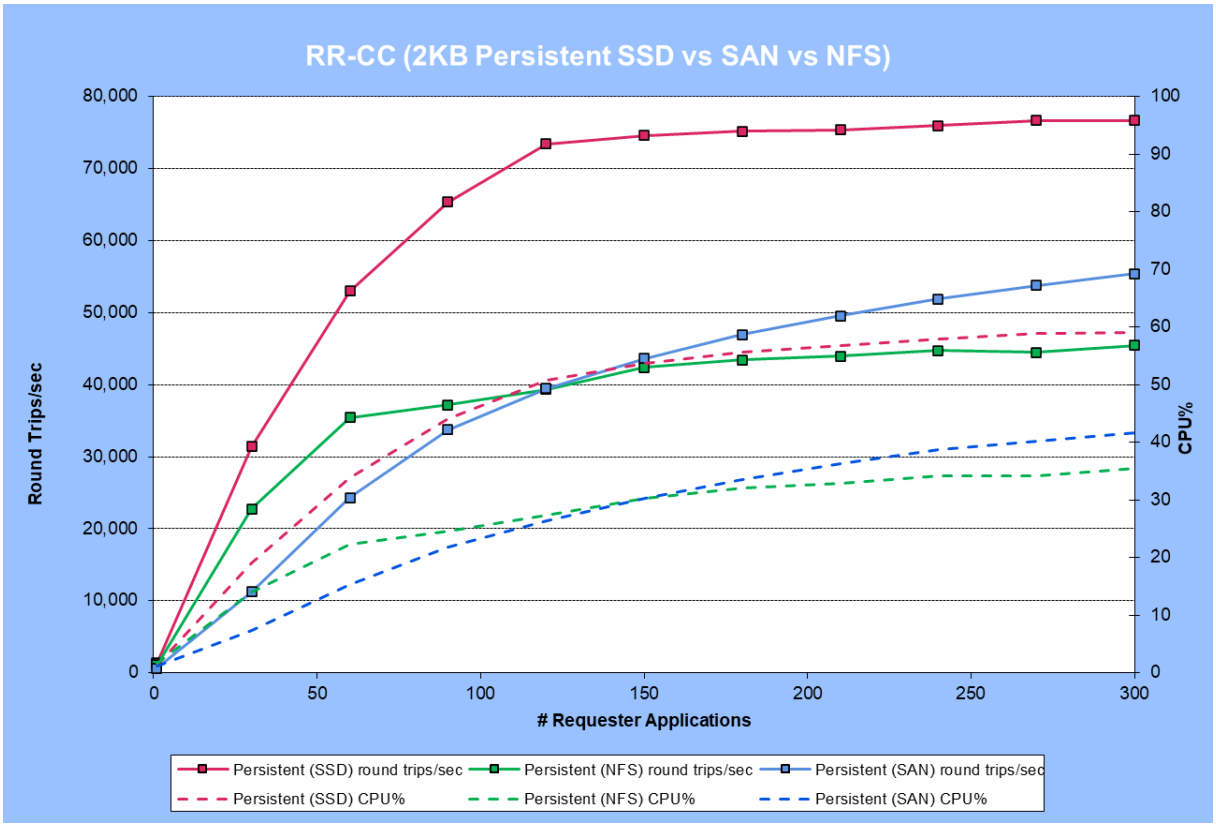

FIGURE 9 - PERFORMANCE RESULTS FOR RR-CC PERSISTENT MESSAGING LOGGING TO SSD, SAN & NFS

To illustrate the impact that the filesystem hosting the recovery logs can have, Figure 9 shows results from running the RR-CC workload with persistent messaging where the recovery logs are on a local SSD or hosted remotely (SAN or NFS).

As expected, logging to a local SSD is a lot faster. The SAN tests are limited by the bandwidth of the SAN switch (16Gb ports). For NFS, the network link is 100Gb, but independent tests showed a limit of around 12Gb/s for single threaded transfers (which the MQ logger must by design perform, to maintain data integrity). The MQ logger will perform larger writes as the number of applications increase but there is a 512KB write size limit for NFS, in AIX V7.3 (which was set for these tests).

Table 8 below, shows the peak rates achieved for each filesystem tested, across a range of message sizes.

Test	V9.4 FP10 Max Rate*	CPU%	Clients
RR-CC (2KB Persistent - SSD)	76,686	59.13	300
RR-CC (2KB Persistent - SAN)	55,424	41.7	300
RR-CC (2KB Persistent - NFS)	45,435	35.4	300
RR-CC (20KB Persistent - SSD)	55,088	49.3	270
RR-CC (20KB Persistent - SAN)	22,116	20.58	300
RR-CC (20KB Persistent - NFS)	10,161	10.68	270
RR-CC (200KB Persistent - SSD)	6,871	16.68	90
RR-CC (200KB Persistent - SAN)	2,691	7.65	135
RR-CC (200KB Persistent - NFS)	1,117	3.58	45
RR-CC (2MB Persistent - SSD)	622	11.65	25
RR-CC (2MB Persistent - SAN)	263	5.3	25
RR-CC (2MB Persistent - NFS)	109	3.38	45

Round trips/ sec

TABLE 8 - TABLE 9 - PEAK RATES FOR WORKLOAD RR-CC (PERSISTENT SSD VS SAN VS NFS)

1.1.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux), NFS Server (x64 Linux) - see section A.1.

For more information on the impact of filesystems, along with some general guidance, on best practises, and monitoring see this (Linux x86) paper.

https://ibm-messaging.github.io/mqperf/mqio_v1.pdf

6 RR-CC Workload with TLS

(Client mode requesters and responders on separate hosts).

To illustrate the overhead of enabling TLS to encrypt traffic between the client applications and the queue manager, results are shown below comparing the performance of the 4 strongest TLS1.2 MQ CipherSpecs, and all TLS1.3 MQ CipherSpecs.

The following TLS 1.2 CipherSpecs were tested (all utilise 256bit encryption and are FIPS compliant).

- ECDHE_ECDSA_AES_256_CBC_SHA384
- ECDHE_ECDSA_AES_256_GCM_SHA384 (Suite B compliant)
- ECDHE_RSA_AES_256_CBC_SHA384
- ECDHE_RSA_AES_256_GCM_SHA384

Results for the suite B compliant CipherSpec (ECDHE_ECDSA_AES_256_GCM_SHA384), along with an older, CBC based CipherSpec (ECDHE_RSA_AES_256_CBC_SHA384) and a TLS 1.3 CipherSpec (TLS_AES_256_GCM_SHA384) are plotted below. As will be seen, the remaining tested CipherSpecs exhibited a performance profile similar to one of these plots.

6.1 TLS Non-Persistent Results

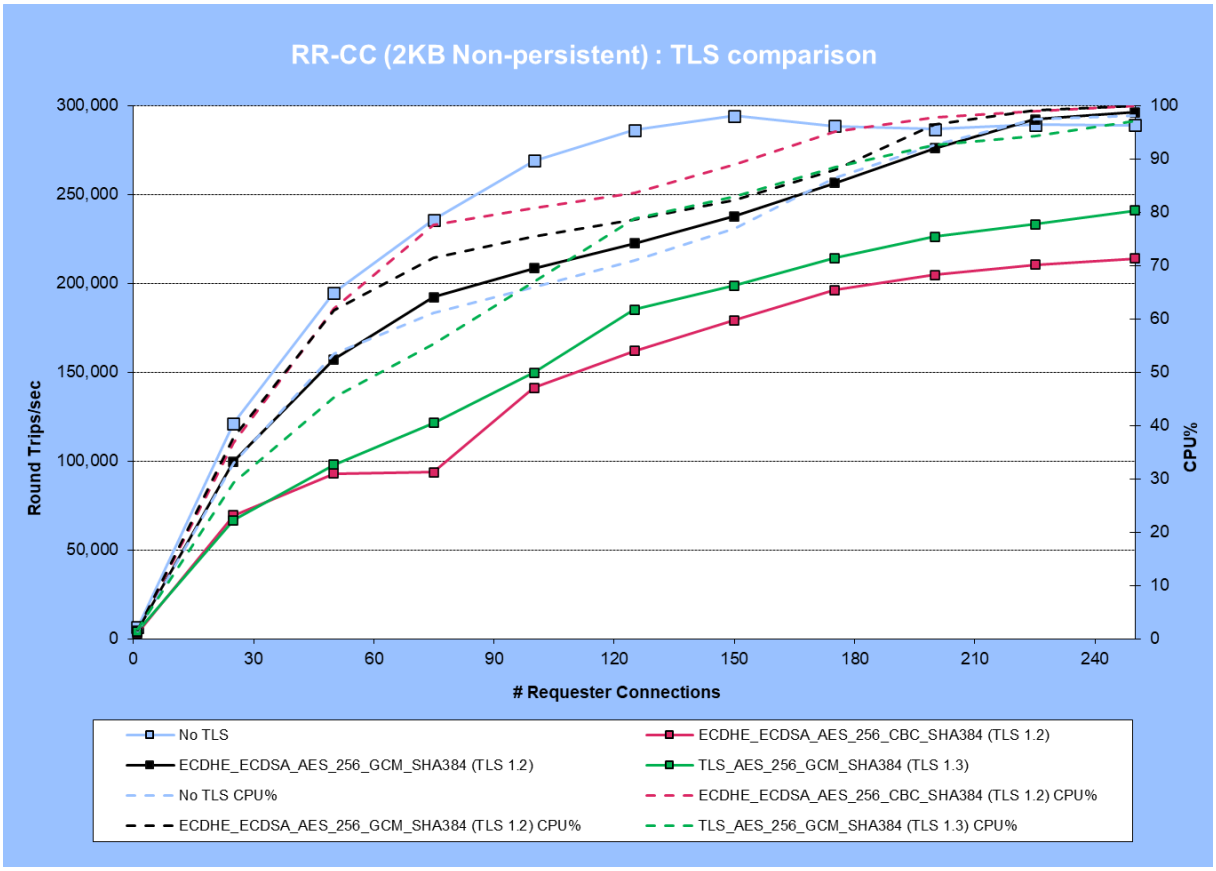

FIGURE 10 - PERFORMANCE RESULTS FOR RR-CC WITH TLS

The ECDHE_ECDSA_AES_256_GCM_SHA384 CipherSpec uses a GCM (Galois/Counter Mode) symmetric cipher. Performance testing showed that all TLS 1.2 GCM based CipherSpecs exhibited similar performance. All the TLS 1.2 CipherSpecs utilising the older CBC (Chain Block Cipher) symmetric cipher exhibited similar to ECDHE_ECDSA_AES_256_CBC_SHA384 in the plot above. All TLS 1.3 CipherSpecs exhibited a performance profile similar to TLS_AES_256_GCM_SHA384 in the plot above.

All tests exhibited scaling up to around 100% of the CPU of the machine. Throughput for GCM based CipherSpecs showed a performance overhead at lower numbers of clients but reached near parity with the non-TLS workload when the number of clients was high, approaching CPU saturation. CBC based CipherSpecs exhibited a greater overhead, running at approximately 74% of a non-encrypted workload. The TLS 1.3 CipherSpec performance sat between the two levels of TLS 1.2 overhead at about 82% of the non-encrypted rate.

Table 9 shows the peak rates achieved for all 6 TLS 1.2 CipherSpecs tested, demonstrating the equivalence of performance, based on whether the symmetric key algorithm is CBC, or GCM based.

TLS 1.2 CipherSpec	V9.4 FP10 Max Rate*	CPU%	Clients
No TLS	294,202	77	150
ECDHE_ECDSA_AES_256_CBC_SHA384	213,861	100	250
ECDHE_ECDSA_AES_256_GCM_SHA384	296,227	100	250
ECDHE_RSA_AES_256_CBC_SHA384	213,464	100	250
ECDHE_RSA_AES_256_GCM_SHA384	294,370	100	250

Round trips/sec

TABLE 9 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB NON-PERSISTENT) – TLS 1.2

Table 10 shows the peak rates achieved for all TLS 1.3 CipherSpecs.

TLS 1.3 CipherSpec	V9.4 FP10 Max Rate*	CPU%	Clients
No TLS	294,202	77	150
TLS_AES_128_CCM_8_SHA256	227,650	100	250
TLS_AES_256_GCM_SHA384	240,891	97	250
TLS_CHACHA20_POLY1305_SHA256	236,777	98	250
TLS_AES_128_GCM_SHA256	241,130	96	250
TLS_AES_128_CCM_SHA256	219,730	100	250

Round trips/sec

TABLE 10 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB NON-PERSISTENT) – TLS 1.3

6.1.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux) - see section A.1.

6.2 TLS Persistent Results

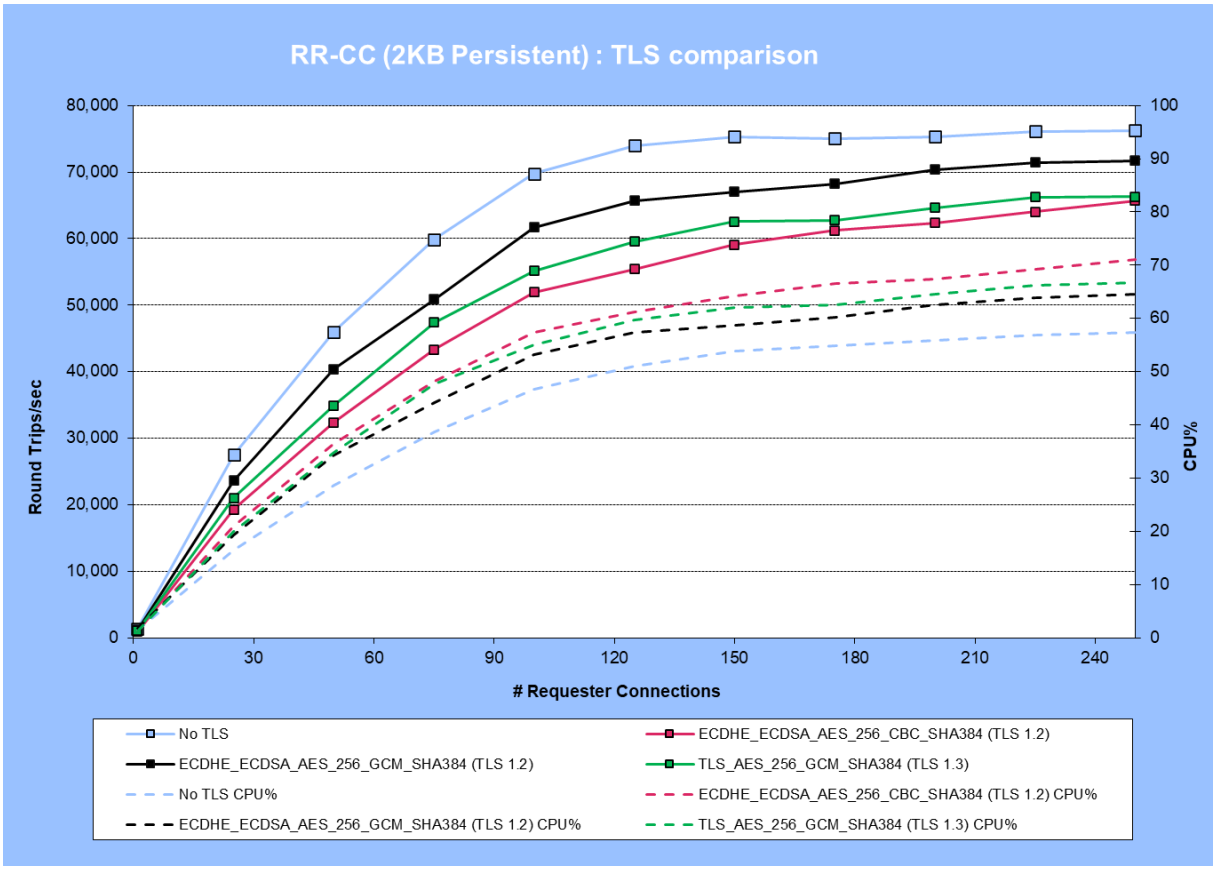

FIGURE 11 - PERFORMANCE RESULTS FOR RR-CC (2KB PERSISTENT) WITH TLS

Generally, the persistent TLS measurements showed a similar pattern to non-persistent. The differences between the performance of the various CipherSpecs is less obvious due to impact of recovery logging.

Table 11 and Table 12 show the peak throughputs for TLS 1.2 & TLS1.3 CipherSpecs.

TLS 1.2 CipherSpec	V9.4 FP10 Max Rate*	CPU%	Clients
No TLS	76,275	57	250
ECDHE_ECDSA_AES_256_CBC_SHA384	65,684	71	250
ECDHE_ECDSA_AES_256_GCM_SHA384	71,693	65	250
ECDHE_RSA_AES_256_CBC_SHA384	65,409	71	250
ECDHE_RSA_AES_256_GCM_SHA384	71,026	64	250

Round trips/sec

TABLE 11 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB PERSISTENT) – TLS 1.2

	V9.4 FP10 Max Rate*	CPU%	Clients
No TLS	76,275	57	250
TLS_AES_128_CCM_8_SHA256	65,879	69	250
TLS_AES_256_GCM_SHA384	66,336	67	250
TLS_CHACHA20_POLY1305_SHA256	67,660	68	250
TLS_AES_128_GCM_SHA256	67,401	67	250
TLS_AES_128_CCM_SHA256	65,167	69	250

Round trips/sec

TABLE 12 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB PERSISTENT) – TLS 1.3

6.2.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux) - see section A.1

6.3 Effect of MQ Recovery Log Performance on TLS Comparisons

With persistent messaging, file I/O to the MQ recovery log is a significant throttling factor. This might show the impact of TLS to be lower in this environment. For TLS 1.3 there is a small improvement in the overhead (expressed as a ratio below) but this is close to the margin of error for the tests. Non-persistent and persistent TLS tests actually exhibited similar TLS overheads. As more work is done outside of the messaging (e.g. in any local applications, these overhead of TLS would be expected to reduce further).

Scenario	**Rate**
No-TLS	294,202
ECDHE_ECDSA_AES_256_GCM_SHA384 (TLS 1.2)	296,227
TLS_AES_256_GCM_SHA384 (TLS 1.3)	240,891

Non-persistent ratio (No-TLS/TLS 1.2) = **0.99**
Non-persistent ratio (No-TLS/TLS 1.3) = **1.22**

For persistent messaging:

Scenario	**Rate**
No-TLS	76,275
ECDHE_ECDSA_AES_256_GCM_SHA384 (TLS 1.2)	71,693
TLS_AES_256_GCM_SHA384 (TLS 1.3)	66,366

Persistent ratio (No-TLS/TLS 1.2) = **1.06**
Persistent ratio (No-TLS/TLS 1.3) = **1.15**

The persistent tests were run with the MQ recovery logs hosted on local, enterprise class NVMe devices, which are very fast. Hosting the recovery log off-box will result in lower throughputs for persistent messaging and the comparison between non-TLS and TLS results will be more favourable (in throughput terms) though the CPU cost will be similar.

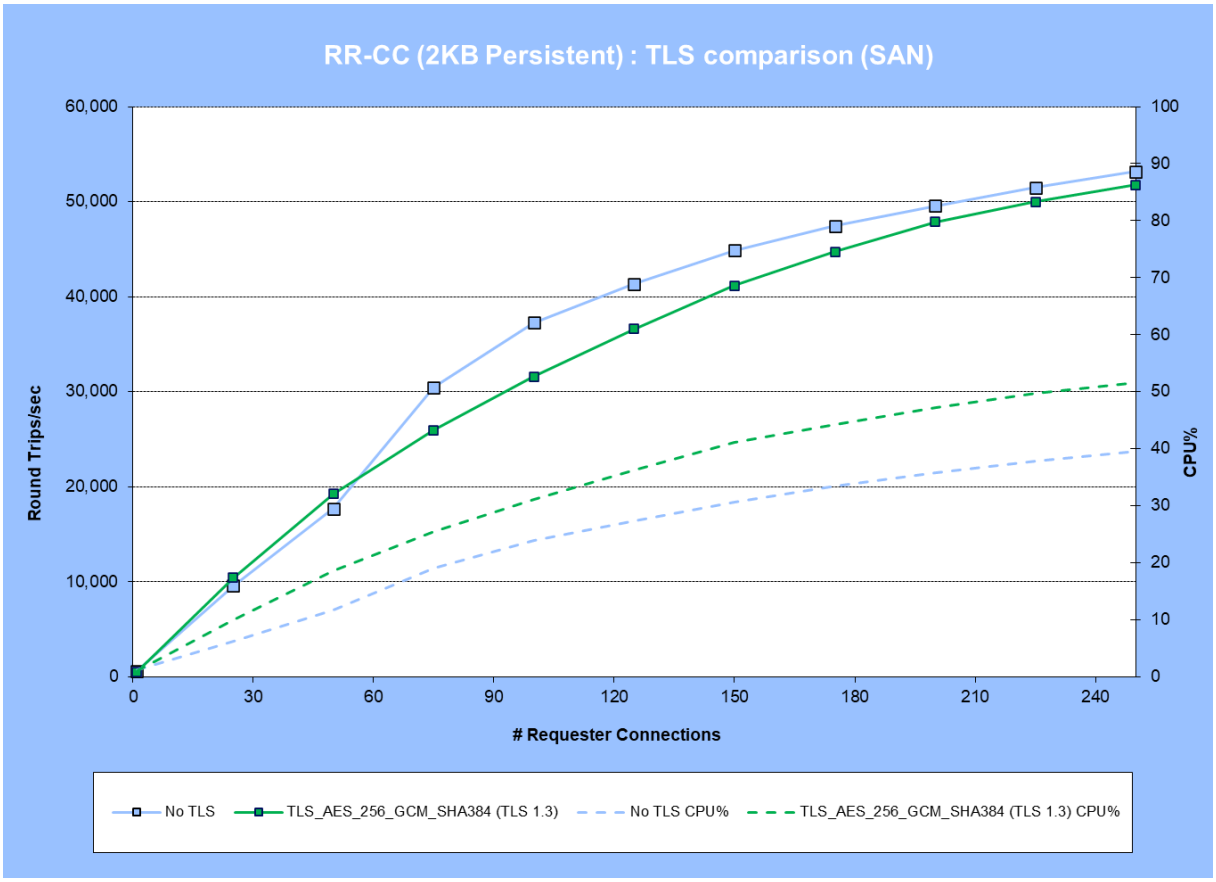

FIGURE 12 - PERFORMANCE RESULTS FOR RR-CC (2KB PERSISTENT) WITH TLS 1.3 (LOGGING TO SAN)

Figure 12 shows results for the RR-CC workload where the MQ recovery log is hosted on a SAN device (see appendix A.1 for details). In this case the throughputs are lower, as the recovery log file writes to the SAN filesystem are slower. As a result the comparison (in throughput terms) between non-TLS and TLS is more favourable (see below).

Scenario	**Rate**
No-TLS	53,161
TLS_AES_256_GCM_SHA384 (TLS 1.3)	51,775

Persistent ratio (No-TLS/TLS 1.3) = **1.03**

Note that the CPU overhead between non-TLS and TLS 1.3 (per round trip) remains similar however. For SAN and local storage, TLS 1.3 CPU was about 1.3 times that for non-TLS, per round-trip (calculated at the rates achieved with 250 clients).

When evaluating TLS, you need to understand the performance capabilities of your infrastructure. Whilst there is a CPU cost incurred with encryption, if you have enough capacity the throughput impact may not be as much as the worst case for persistent message, as shown in Figure 11.

As for all performance evaluations, testing an environment as close as possible to that used in production is highly desirable. This will result in a much better understanding of the performance capabilities of the various components making up the environment your workload is running in.

6.3.1 Test setup

Workload type: RR-CC (see section 3.1).

Hardware: Server 1 (AIX), Client 2 (x64 Linux), Client 3 (x64 Linux) - see section A.1

Appendix A: Test Configurations

A.1 Hardware/Software – Set1

All the testing in this document (apart from when testing results are shown from a different platform and are clearly identified as such) was performed on the following hardware and software configuration:

A.1.1 Hardware

Server1 (AIX on Power 10 Machine)

- IBM Power S1022 (9105-22A)
- 2CPU x 16-Cores (SMT8) 2.75 - 4.0 GHz (max)
    - Configured as SMT4 (optimal for typical MQ workloads)
- 2 x 1.6TB NVMe 4K Gen4 U.2 (ES4B) Enterprise SSDs (hosting queue manager data and log filesystems).
- 2 x 800GB NVMe Gen4 U.2 Slim SSD
- 512GB RAM
- 100Gb network adapter on an isolated performance LAN.

Client1 (AIX on Power 10 Machine)

- IBM Power S1022 (9105-22A)
- 2CPU x 16-Cores (SMT8) 2.75 - 4.0 GHz (max)
    - Configured as SMT4 (optimal for typical MQ workloads)
- 2 x 1.6TB NVMe Gen4 U.2 SSDs
- 2 x 800GB NVMe Gen4 U.2 Slim SSD
- 512GB RAM
- 100Gb network adapter on an isolated performance LAN.

Client2

- ThinkSystem SR630 V3, Model:7D73CTO1WW
- 2 x 16 core INTEL(R) XEON(R) GOLD 6544Y CPU @3.60GHz
- 2 x LENOVO 03GX685 - ThinkSystem 2.5in PM1655 800GB Mixed Use SAS 24Gb HS SSDs.
- 2 x 1.6TB NVMe SSD devices configured as a RAID0 array.
- 100Gb network adapter on an isolated performance LAN.
- Hyper-Threading is enabled but Turbo Boost is disabled. This is to assist with achieving the best performance that is also consistent.

Client3

- ThinkSystem SR630 V3, Model:7D73CTO1WW
- 2 x 16 core INTEL(R) XEON(R) GOLD 6544Y CPU @3.60GHz
- 2 x LENOVO 03GX685 - ThinkSystem 2.5in PM1655 800GB Mixed Use SAS 24Gb HS SSDs.
- 2 x 1.6TB NVMe SSD devices configured as a RAID0 array.
- 100Gb network adapter on an isolated performance LAN.
- Hyper-Threading is enabled but Turbo Boost is disabled. This is to assist with achieving the best performance that is also consistent.

NFS Server:

- ThinkSystem SR630 V2– [7Z71CTO1WW]
- 2 x 16 core CPUs.
  Core: Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz
- 256GB RAM
- 2x3TB NVMe drives configured as a RAID0 array hosting NFS exported directories.
- 100Gb ethernet adapter on an isolated performance LAN.
- Hyper-Threading is enabled but Turbo Boost is disabled. This is to assist with achieving the best performance that is also consistent.

A.1.2 Software (AIX Hosts)

- IBM AIX V7.3.0.0 TL03 SP00
- MQ-CPH MQI test driver (see Appendix C:)
- IBM MQ V9.4.0.10 (V9.4 with Fix Pack 10)

A.1.3 Software (Linux Hosts)

- RedHat Enterprise Linux V9.5 (Plow)
- MQ-CPH MQI test driver (see Appendix C:)
- JMSPerfharness test driver (see Appendix C:)
- IBM MQ V9.4.0.10 (V9.4 with Fix Pack 10)

A.2 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.2.1 Operating System

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

Malloc environment variables:

```
MALLOCOPTIONS=multiheap
MALLOCTYPE=buckets
```

100Gb network settings overriding system defaults:

```
tcp_nodelay   1            Enable/Disable TCP_NODELAY Option
tcp_recvspace 1048576      Set Socket Buffer Space for Receiving
tcp_sendspace 1048576      Set Socket Buffer Space for Sending
```

Network Settings (system defaults):

```
rfc1323 = 1
sb_max = 1300000
```

User limits

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max memory size         (kbytes, -m) 32768
open files              (-n) 100000
pipe size               (512 bytes, -p) 64
stack size              (kbytes, -s) 4194304
cpu time                (seconds, -t) unlimited
max user processes      (-u) 4096
virtual memory          (kbytes, -v) unlimited
```

NFS

NFS mount for the MQ recovery log in NFS tests used the following parameters:
`rsize=524288,wsize=524288`

A.2.2 IBM MQ

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

```
Channels:
   MQIBindType=FASTPATH
   MaxActiveChannels=5000
   MaxChannels=5000

Log:
   LogBufferPages=4096
   LogFilePages=32767
   LogPrimaryFiles=64
   LogSecondaryFiles=2
   LogType=CIRCULAR
   LogWriteIntegrity=TripleWrite

TuningParameters:
   DefaultPQBufferSize=10485760
   DefaultQBufferSize=10485760
```

For large message sizes (200K & 2MB), the queue buffers were increased further to:

```
DefaultPQBufferSize=104857600
DefaultQBufferSize=104857600
```

Note that large queue buffers may not be needed in your configuration. Writes to the queue files are asynchronous, taking advantage of OS buffering. Large buffers were set in the runs here, as a precaution only.

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

Appendix B: Glossary of terms used in this report

CD	Continuous delivery.
JMSPerfharness	JMS based, performance test application (https://github.com/ot4i/perf-harness)
LTS	Long term service.
MQ-CPH	C based, performance test application (https://github.com/ibm-messaging/mq-cph)

Appendix C: Resources

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

IBM MQ Performance: Best Practises, and Tuning Paper:
https://ibm-messaging.github.io/mqperf/MQ_Performance_Best_Practices_v1.0.1.pdf

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

Tutorial:

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

Tutorial: