

IBM MQ V10.0 for Linux (x86-64 platform) Performance Report

Version 1.0 - October 2026

Aneesha Yerrathota
IBM MQ Performance
IBM India Laboratories
Bangalore
India

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, September 2026.
This edition applies to *IBM MQ V10.0* (and to all subsequent releases and modifications until otherwise indicated in new editions).
© Copyright International Business Machines Corporation 2026. All rights reserved.

Note to U.S. Government Users
Documentation related to restricted rights.
Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

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

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

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

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

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

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

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

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

INTENDED AUDIENCE
This report is intended for architects, systems programmers, analysts and programmers wanting to understand the performance characteristics of IBM MQ V10.0. 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 V10.0.

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 V10.0 for Linux on x86_64.
- Want to understand the performance limits of IBM MQ V10.0 for Linux on x86_64.
- Want to understand what actions may be taken to tune IBM MQ V10.0 for Linux on x86_64.

The reader should have a general awareness of the Linux 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: MQ_Performance_Best_Practices_v1.0.2.pdf

Contents

This report includes:

- 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 smassey@uk.ibm.com & aneesha.yerrathota@ibm.com

Contents

Preface....5
1 Introduction....8
2 Release Highlights....9
2.1 Change to default TCP buffer size for IBM MQ clients....9
2.1.1 Test setup.....9
2.2 Customizing JMS connection keep alive behavior....10
2.3 Changes to CipherSuites supported by the IBM MQ....10
2.4 Quantum-safe key....11
2.4.1 Test setup.....12
3 Base MQ Performance Workloads....12
3.1 RR-CC Workload....14
3.2 RR-DQ-BB Workload....15
4 Non-Persistent Performance Test Results....16
4.1 RR-CC Workload....16
4.1.1 Test setup.....17
4.2 RR-DQ-BB Workload....17
4.2.1 Test setup.....18
4.3 RR-CC JMS Workload....18
4.3.1 Test setup.....19
5 Persistent Performance Test Results....20
5.1 RR-CC Workload....20
5.1.1 Test setup.....21
5.2 Impact of Different File Systems on Persistent Messaging Performance....21
5.2.1 Test setup.....23
6 RR-CC Workload with TLS....23
6.1 TLS Non-Persistent Results....24
6.1.1 Test setup.....25
6.2 TLS Persistent Results....25
6.2.1 Test setup.....26
Appendix A: Test Configurations....27
A.1 Hardware/Software – Set1....27
A.1.1 Hardware....27
A.1.2 Software....27
A.1.3 Software....27
A.2 Tuning Parameters Set for Measurements in This Report....27
A.2.1 Operating System....28
A.2.2 IBM MQ....28
Appendix B: Glossary of terms used in this report.....29
Appendix C: Resources....30

TABLES

Table 1- QUANTUM-SAFE KEY EXCHANGE WORKLOAD 12
Table 2 - Workload types 13
Table 3 - Peak rates for workload RR-CC (non-persistent) 17
Table 4 – Full Results for workload RR-DQ-BB (non-persistent) 18
Table 5 - Peak rates for JMS (non-persistent) 19
Table 6 - Peak rates for workload RR-CC (non-persistent) 21
Table 7 - Peak rates for workload RR-CC (Persistent) 21
Table 8 - Peak rates for workload RR-CC (Persistent SSD vs NFS) 22
Table 9 - Peak rates for MQI client bindings (2KB non-persistent) – TLS 1.2 24
Table 10 - Peak rates for MQI client bindings (2KB non-persistent) – TLS 1.3 25
Table 11 - Peak rates for MQI client bindings (2KB persistent) – TLS 1.2 26
Table 12 - Peak rates for MQI client bindings (2KB persistent) – TLS 1.3 26

FIGURES

Figure 1- Performance results for RR-CC (200KB JMS non-persistent) 9
Figure 2- QUANTUM-SAFE KEY EXCHANGE RESULTS 12
Figure 3 - RR-CC Topology 14
Figure 4 - Requester-responder with remote queue manager (remote responders) 15
Figure 5 - Performance results for RR-CC (2KB non-persistent) 16
Figure 6 - Performance results for RR-DQ-BB (2KB non-persistent) 17
Figure 7 - Performance results for RR-CC (2KB JMS non-persistent) 18
Figure 8 - Performance results for RR-CC (2KB Non-persistent vs Persistent) 20
Figure 9 - Performance Results for RR-CC Persistent Messaging logging to SSD & NFS 22
Figure 10 - Performance Results for RR-CC with TLS 24
Figure 11 - Performance Results for RR-CC (2KB Persistent) with TLS 25

1 Introduction

IBM MQ V10.0 is a long-term service (LTS) release of MQ, which includes features made available in the V9.4.1, V9.4.2, V9.4.3, V9.4.4 & V9.4.5 continuous delivery (CD) releases.

Performance data presented in this report does not include release to release comparisons, but all tests run showed equal or better performance than the V9.4 release of IBM MQ.

The hardware used in the report is newer than that used in the MQ V9.4 LTS performance report, The machines have been upgraded to ThinkSystem SR630 V3 with Intel Xeon Gold 6544Y processors. In all cases the version of Red Hat Linux is newer (RHEL 9.8) and all security patching between the release of MQ V9.4 and MQ V10.0 has been applied. As such, direct numerical comparisons with the V9.4 report should be made with caution, as both hardware and OS differences will influence the results.

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 10.0 documentation here:
https://www.ibm.com/docs/en/ibm-mq/10.0.x?topic=mq-whats-new-changed-in-1000

2.1 Change to default TCP buffer size for IBM MQ clients

Previously, default TCP buffer sizes differed between clients. C clients used the operating system's default, while the JMS, XMS .NET and .NET clients used about 32 KB. To get the OS default on those clients, you had to set ClntRcvBuffSize and ClntSndBuffSize to zero.

From IBM MQ 10.0.0, all clients behave the same way. When a client creates a TCP connection, it checks the current buffer size:

- If the size is below 32768 bytes, the client requests 32768 bytes.
- If the size is 32768 bytes or more, the client lets the network layer decide.

This generally gives higher message throughput as it allows the OS to dynamically configure TCP buffer sizes to cater for a range of network bandwidths and latencies.
See ClntRcvBuffSize and ClntSndBuffSize in the TCP stanza of the client configuration file.

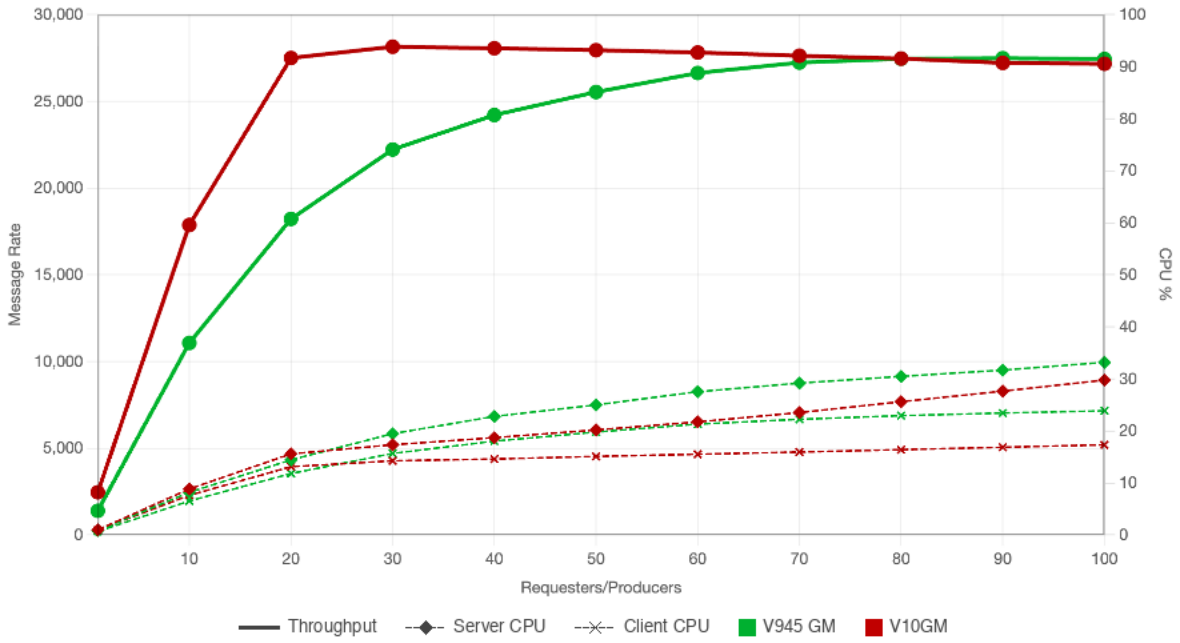


FIGURE 1- PERFORMANCE RESULTS FOR RR-CC (200KB JMS NON-PERSISTENT)

2.1.1 Test setup.

Workload type: RR-CC (see section 3.1).
Message protocol: JMS.
Hardware: Server 1, Client 1, Client 2 (see section A.1).

2.2 Customizing JMS connection keep alive behavior

From IBM MQ 10.0.0, you can customize the keep alive behavior of a JMS connection to avoid unexpected disconnections that might occur when the queue manager enforces a channel disconnect interval (DISCINT).

If a JMS or Jakarta Messaging connection uses a channel with DISCINT greater than 0, the queue manager can disconnect that channel for lack of activity. The application then reports JMSExceptions containing IBM MQ reason code 2009(MQRC_CONNECTION_BROKEN).

From IBM MQ 10.0.0, a new mechanism keeps the channel instances used by JMS or Jakarta Messaging connections open, so they are not closed due to DISCINT. This improves the stability of the application.

Default behavior

By default, the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging periodically issue an MQSTAT API call for each JMS connection to keep all associated JMS sessions active. The interval is determined by the channel attribute HBINT. If HBINT=0, a default interval of 300 seconds is used.

Customizing the interval

Override the default interval by running your application with -Dcom.ibm.mq.jms.tuning.connectionKeepAlive=<value>, where <value> is:

- A positive integer defining the required disconnect interval in seconds.
  Recommended: half of the channel's DISCINT value, or half of the smallest DISCINT value if the application connects over multiple channels.
- -1 to disable the JMS connection keep alive flow.
- 0 or any other negative value (except -1) is ignored, and the interval is derived from HBINT.

Applicability

This feature is available only for IBM MQ 10.0.0 (and later) IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging connecting to a queue manager at IBM MQ 9.0 or later.

Verifying keep alive calls

To log each MQSTAT call in the mqjms.log file, run your application with -Dcom.ibm.mq.jms.tuning.connectionKeepAlive.logMQSTAT=true. The default value is false. To read more about this behaviour please refer to IBM Documentation.

2.3 Changes to CipherSuites supported by the IBM MQ

From IBM MQ 10.0.0, the following CipherSuites are not supported for the IBM Semeru Runtime 25 used to run the IBM MQ AMQP service, Managed File Transfer, IBM MQ Console, REST API, and IBM MQ Telemetry server:

- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_NULL_SHA
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_NULL_SHA
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- TLS_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_SHA

If you use any of these components, you must ensure that they use a CipherSuite supported by IBM Semeru Runtime 25.

2.4 Quantum-safe key

From IBM MQ 10.0, support for FIPS 203 ML-KEM algorithms has been added to Linux C client and queue managers. Using the FIPS 203 ML-KEM algorithms enables you to protect your IBM MQ deployments from future quantum computers and the current "Harvest now, Decrypt later" attacks.

IBM MQ supports both ML-KEM and Hybrid ML-KEM key exchange algorithms. For more information about the key shares supported by IBM MQ, see TLS key shares in IBM MQ see TLS key shares in IBM MQ.

To secure existing TLS channels against a quantum computers, TLS channels must have quantum safe algorithms enabled. On outbound channels (or client applications) this is achieved by explicitly setting the use of a quantum safe algorithm. On inbound channels, quantum safe algorithms are always supported unless disabled through the

- Set SSL:AllowedQuantumSafeAlgorithms in the qm.ini file.
- Set SSLQSR(REQUIRED) on the queue manager's SVRCONN channel, with a TLS 1.3 CipherSpec such as TLS_AES_128_GCM_SHA256.
  Set SSLCAUTH(OPTIONAL) to allow anonymous TLS.
- Set SSLQS on the client's CLNTCONN channel, for example SSLQS(HYBMLKEM). Use a CCDT to give the C client the algorithm. In a JSON CCDT the setting is transmissionSecurity:quantumSafeAlgorithm.
  Alternatively, set the QuantumSafeProtection field on the MQCD structure.

```
DEFINE CHANNEL(MYCHANNEL) CHLTYPE(CLNTCONN) CONNAME('localhost(1414)')
SSLCIPH(TLS_AES_128_GCM_SHA256) SSLQS(HYBMLKEM)
DEFINE CHANNEL(MYCHANNEL) CHLTYPE(SVRCONN) SSLCIPH(TLS_AES_128_GCM_SHA256)
SSLQSR(REQUIRED) SSLCAUTH(OPTIONAL)
```

Report test name	CLNTCONN (*SSLQS*)	SVRCONN (*SSLQSR*)	qm.ini (*SSL:AllowedQuantumSafeAlgorithms*)
PQ = 0 - Baseline (no quantum-safe)	NONE	OPTIONAL	NONE
PQ = 1	HYBMLKEM	REQUIRED	ALL
PQ = 2	MLKEM	REQUIRED	ALL

TABLE 1- QUANTUM-SAFE KEY EXCHANGE WORKLOAD

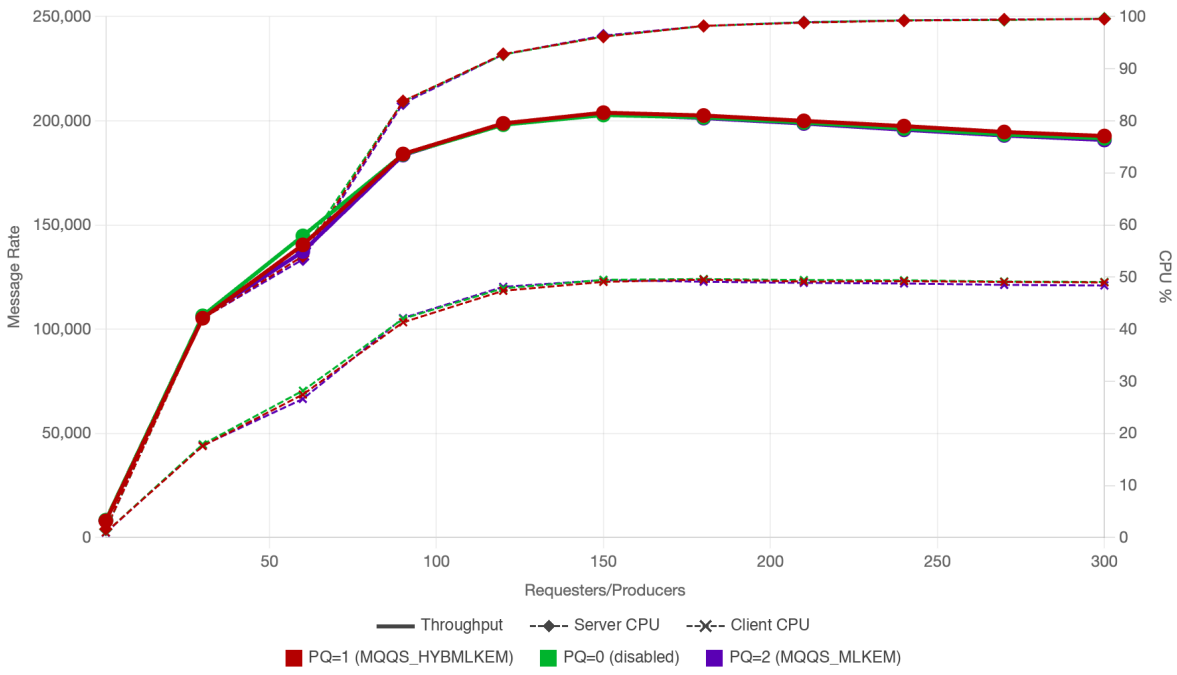


FIGURE 2- QUANTUM-SAFE KEY EXCHANGE RESULTS

Enabling PQ algorithms has no additional cost: PQ=1 (hybrid ML-KEM) and PQ=2 (ML-KEM) perform similarly to the PQ=0 baseline.

2.4.1 Test setup.

Workload type: RR-CC (see section 3.1).
Hardware: Server 1,Client 1, Client 2 (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-CC & RR-DQ-BB are described in the following section.

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.)

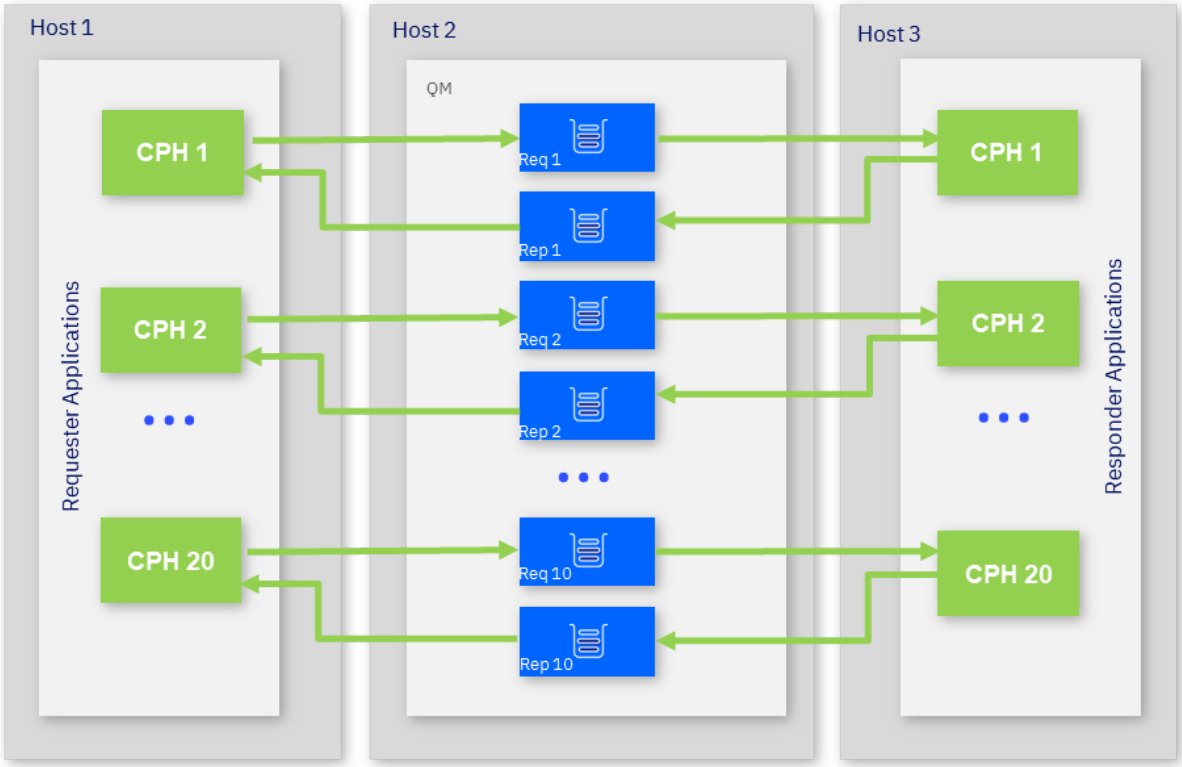


FIGURE 3 - RR-CC TOPOLOGY

Error! Reference source not found.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.
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).

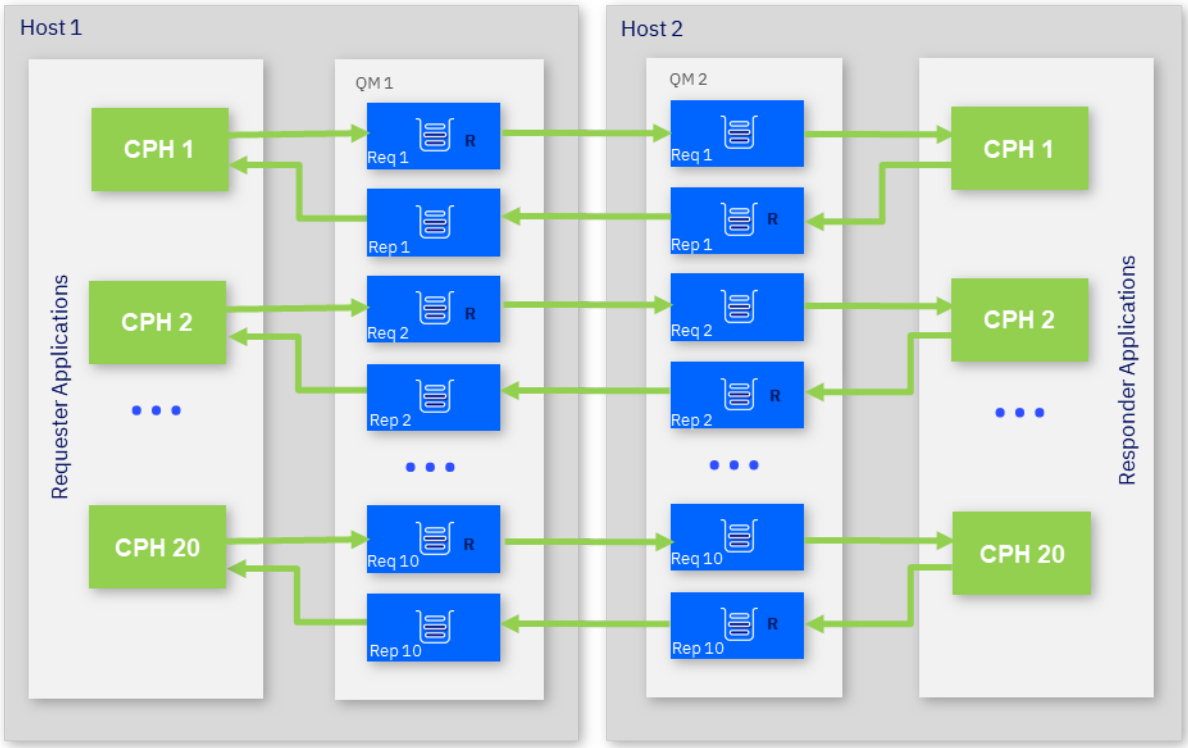


FIGURE 4 - REQUESTER-RESPONDER WITH REMOTE QUEUE MANAGER (REMOTE RESPONDERS)

This is a distributed queuing version of the requester-responder topology detailed in section 3.1. 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.

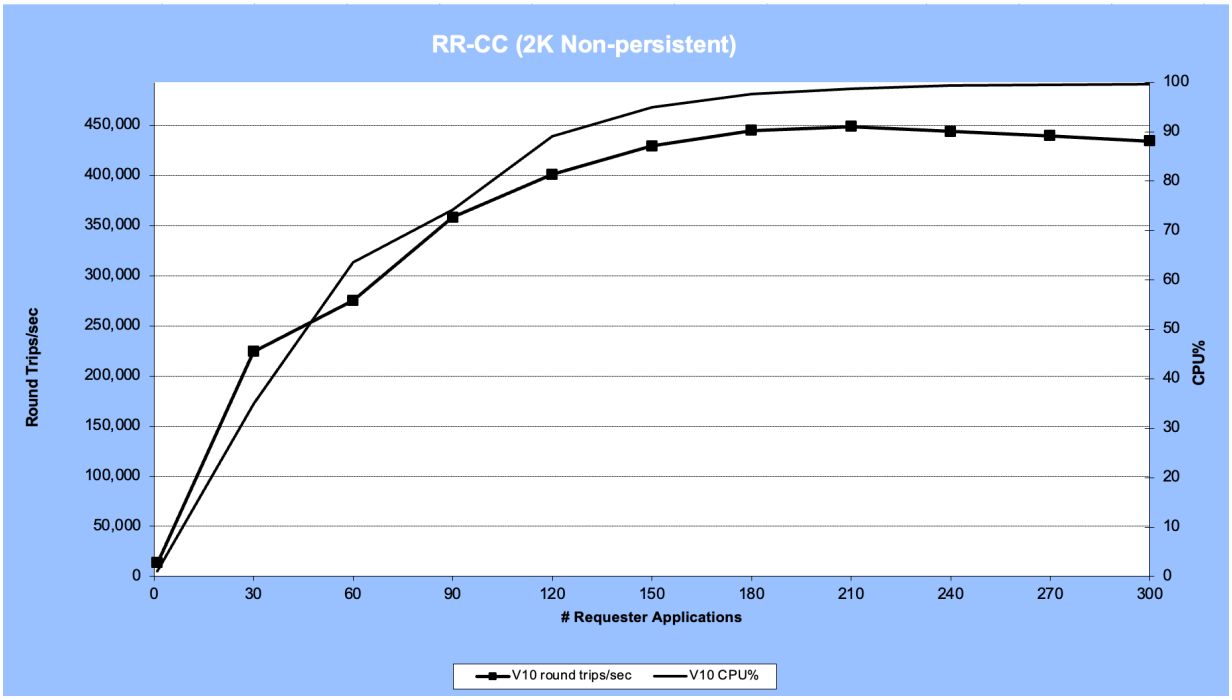


FIGURE 5 - PERFORMANCE RESULTS FOR RR-CC (2KB NON-PERSISTENT)

The test peaked at approximately 448,946 round trips/sec, approaching full CPU utilisation of the MQ server. Peak round trip rates for all message sizes tested can be seen in the table below. The 200KB and 2MB scenarios are approaching saturation of the 100Gb network links between the client and server machines (hence lower CPU% for the highest message rate).

Test	V10 Max Rate*	CPU%	Clients
RR-CC (2K Non-persistent)	448,946	98.71	210
RR-CC (20K Non-persistent)	271,199	75.5	120
RR-CC (200K Non-persistent)	28,483	14.82	20
RR-CC (2MB Non-persistent)	2,160	20.78	40

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,Client 1, Client 2 (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 0) 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.

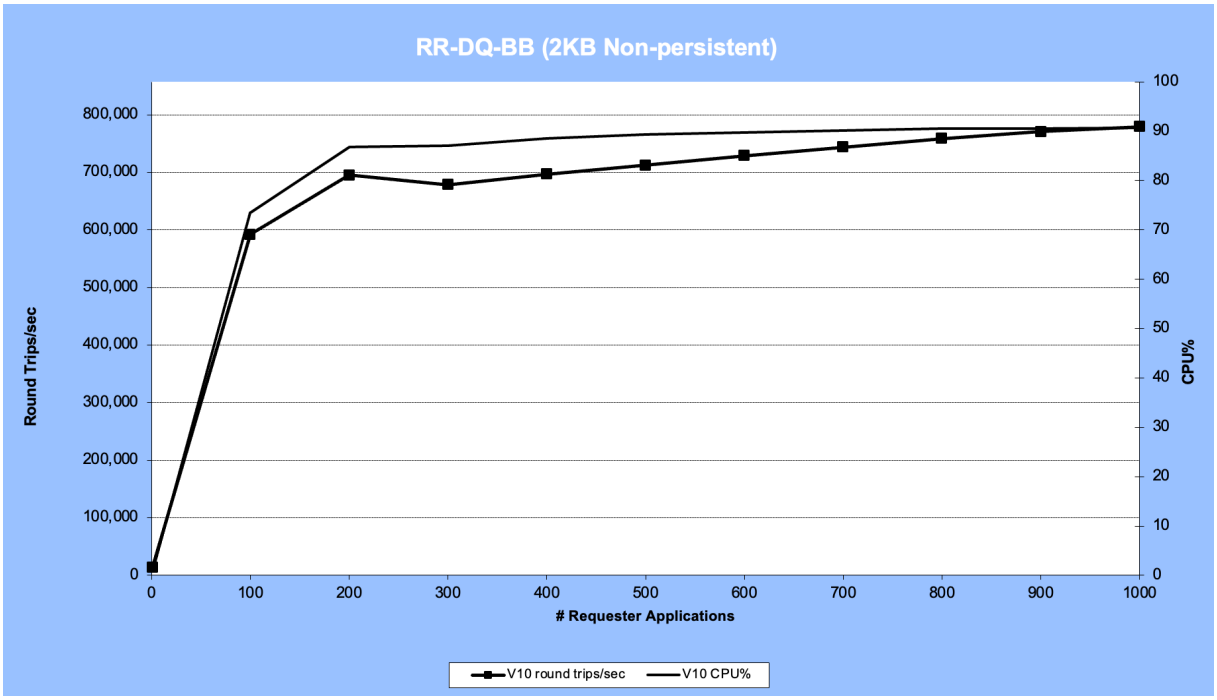


FIGURE 6 - PERFORMANCE RESULTS FOR RR-DQ-BB (2KB NON-PERSISTENT)

The distributed queuing test exhibits good scaling with CPU being the limiting factor as the number of clients increases.

Peak round trip rates for all message sizes tested can be seen in the table below. The 200KB and 2MB measurements are again network limited by the 100Gb links between the hosts. 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 0)

Test	V10 Max Rate*	CPU%	Clients
RR-DQ-BB (2KB Non-persistent)	779,374	90.73	1000
RR-DQ-BB (20KB Non-persistent)	397,982	66.61	120
RR-DQ-BB (200KB Non-persistent)	57,148	20.56	35
RR-DQ-BB (2MB Non-persistent)	5,598	23.38	40

***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 0).
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.

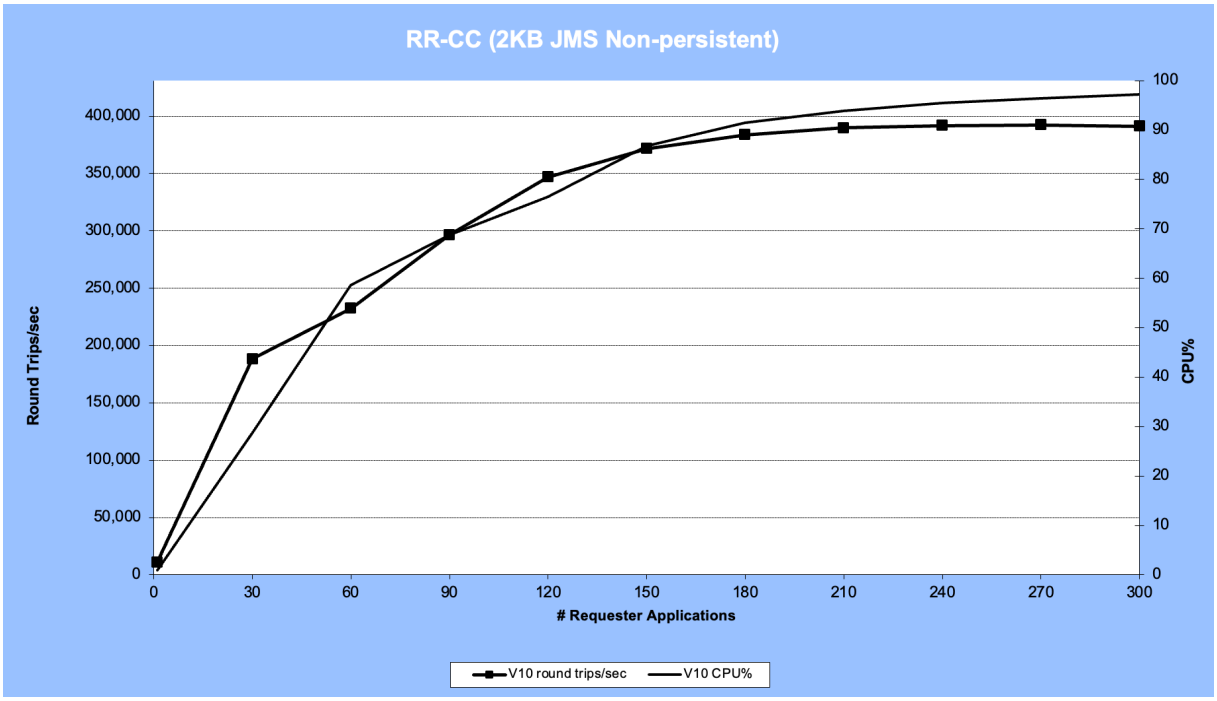


FIGURE 7 - PERFORMANCE RESULTS FOR RR-CC (2KB JMS NON-PERSISTENT)

Once again, the workload exhibits good scaling up to 100% of the CPU (the limiting factor), peaking at approximately 392,000 round trips/sec.

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

Test	V10 Max Rate*	CPU%	Clients
RR-CC (2KB JMS Non-persistent)	392,572	96.44	270
RR-CC (20KB JMS Non-persistent)	267,961	79.69	160
RR-CC (200KB JMS Non-persistent)	28,040	18.13	40
RR-CC (2MB JMS Non-persistent)	2,494	46.44	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 1, Client 2 (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 V10.0 in a separate environment (x64 Linux with SSD & NFS filesystems) are shown to demonstrate the impact of recovery log location.

5.1 RR-CC Workload

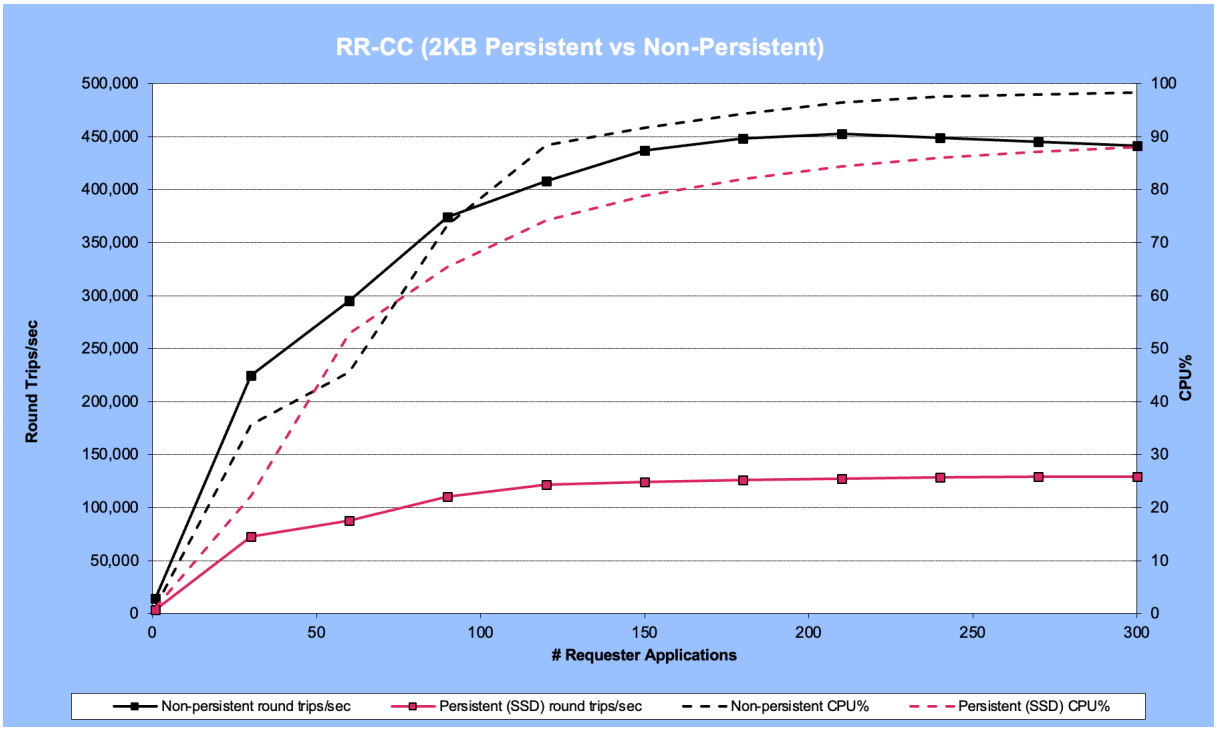


FIGURE 8 - PERFORMANCE RESULTS FOR RR-CC (2KB NON-PERSISTENT VS PERSISTENT)

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 210 requesters where, 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	V10 Max Rate*	CPU%	Clients
RR-CC (2K Non-persistent)	448,946	98.71	210
RR-CC (20K Non-persistent)	271,199	75.5	120
RR-CC (200K Non-persistent)	28,483	14.82	20
RR-CC (2MB Non-persistent)	2,160	20.78	40

**ROUND TRIPS/ SEC*

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

Test	V10 Max Rate*	CPU%	Clients
RR-CC (2KB Persistent)	129,449	88.04	300
RR-CC (20KB Persistent)	105,742	80.04	250
RR-CC (200KB Persistent)	13,595	18.11	60
RR-CC (2MB Persistent)	1,710	20.27	45

**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 (see section A.1).

5.2 Impact of Different File Systems on Persistent Messaging Performance

A separate paper has been published, with illustrative results, for SSD, SAN and NFS hosted filesystems, along with some guidance, on best practises, and monitoring.
https://ibm-messaging.github.io/mqperf/mqio_v1.pdf
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.

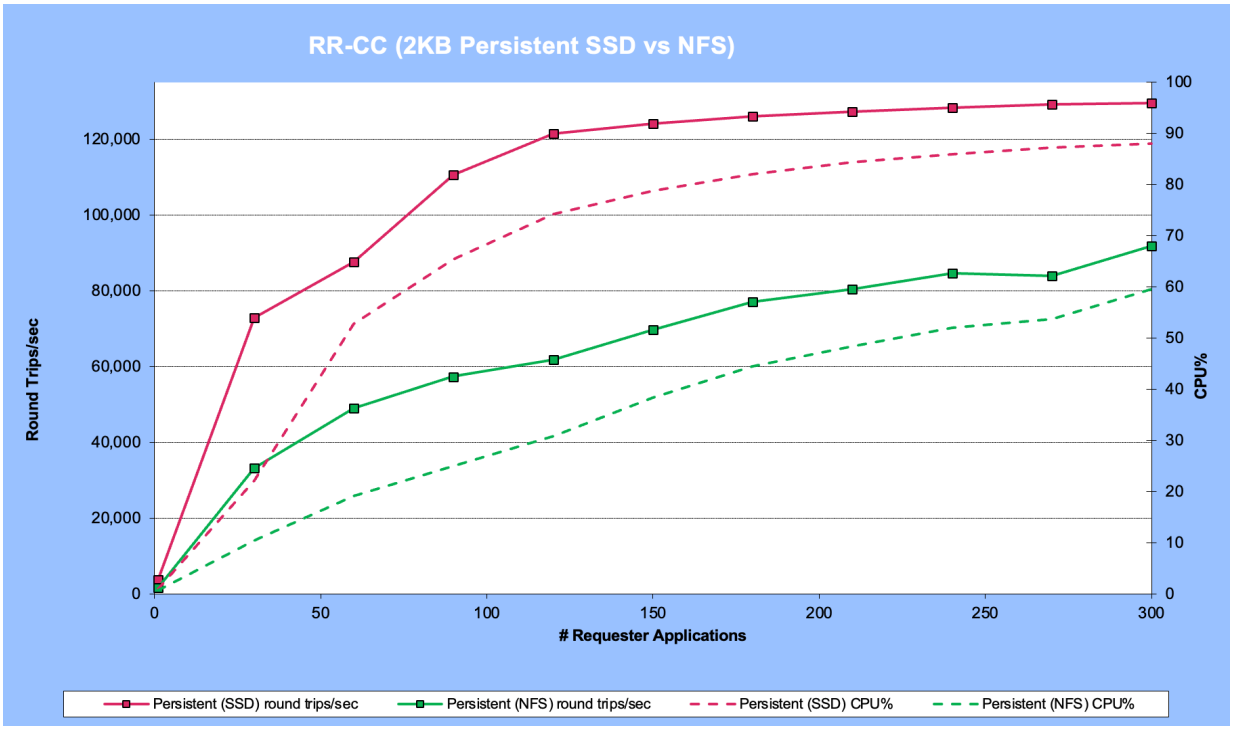


FIGURE 9 - PERFORMANCE RESULTS FOR RR-CC PERSISTENT MESSAGING LOGGING TO SSD & NFS

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 (NFS).
As expected, logging to a local SSD is a lot faster. For NFS, the network link is 100Gb, but independent tests showed a limit of around 27Gb/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 1MB write size limit for NFS, in the Linux kernel.

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

Test	V10 Max Rate*	CPU%	Clients
RR-CC (2KB Persistent)	129,449	88.04	300
RR-CC (2KB Persistent - NFS)	91,785	59.6	300
RR-CC (20KB Persistent)	105,742	80.04	250
RR-CC (20KB Persistent - NFS)	27,585	22.73	300
RR-CC (200KB Persistent)	13,595	18.11	60
RR-CC (200KB Persistent - NFS)	3,349	7.32	135
RR-CC (2MB Persistent)	1,710	20.27	45
RR-CC (2MB Persistent - NFS)	341	4.4	40

Round trips/ sec

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

5.2.1 Test setup.

Workload type: RR-CC (see section 3.1).
Hardware: Server 1, with client 1 machine acting as NFS server. (see section A.1).

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.
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_128_CCM_8_SHA256) 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

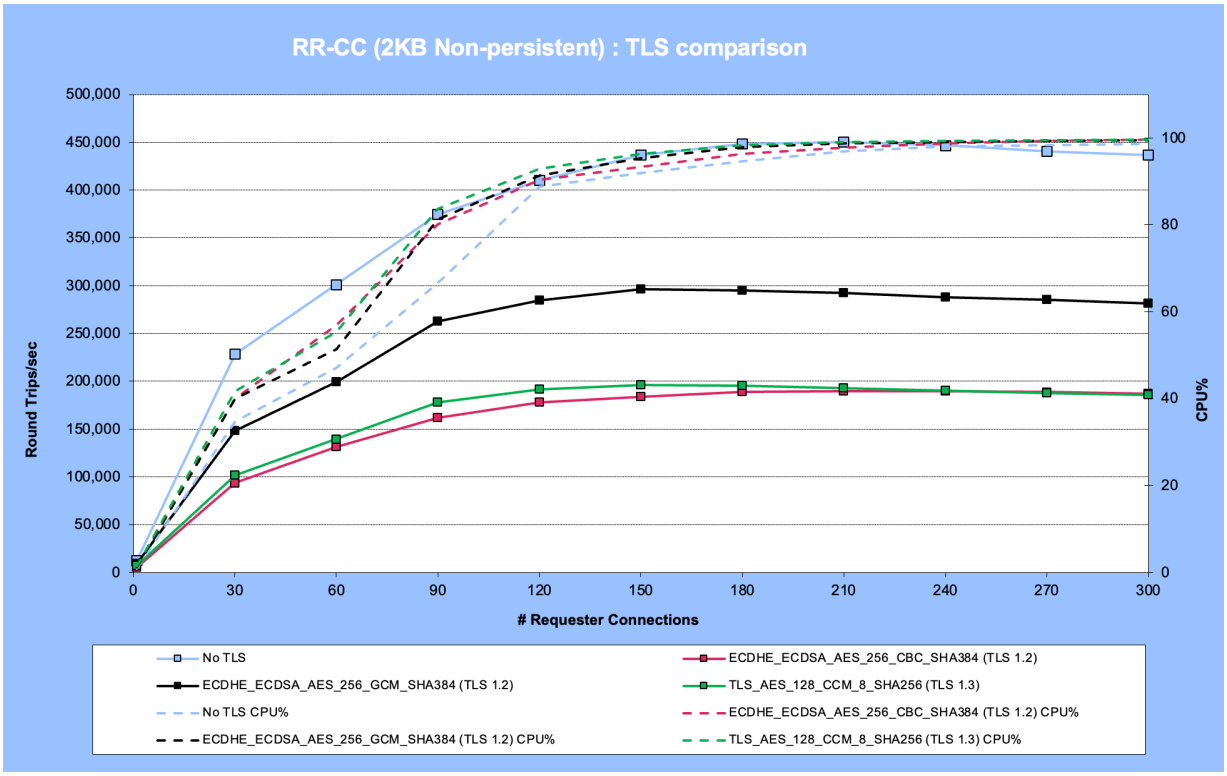


FIGURE 10 - PERFORMANCE RESULTS FOR RR-CC WITH TLS

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_128_CCM_8_SHA256 in the plot above.

All tests exhibited scaling up to around 100% of the CPU of the machine. Throughput for GCM based CipherSpecs ran at approximately 65% of the throughput of a non-encrypted workload. CBC based CipherSpecs exhibited a greater overhead, running at approximately 42% of a non-encrypted workload. TLS 1.3 CipherSpecs achieved rates similar to, and in some cases slightly above, the TLS 1.2 CBC based CipherSpecs.

Table 9 shows the peak rates achieved for all 4 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	V10.0 GM Max Rate*	CPU%	Clients
No TLS	450,331	97	210
ECDHE_ECDSA_AES_256_CBC_SHA384	189,607	99	240
ECDHE_ECDSA_AES_256_GCM_SHA384	294,895	95	150
ECDHE_RSA_AES_256_CBC_SHA384	189,897	98	210
ECDHE_RSA_AES_256_GCM_SHA384	296,404	95	150

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	V10.0 GM Max Rate*	CPU%	Clients
No TLS	450,331	97	210
TLS_AES_128_CCM_8_SHA256	195,267	96	150
TLS_AES_256_GCM_SHA384	203,447	96	150
TLS_CHACHA20_POLY1305_SHA256	196,337	96	150
TLS_AES_128_GCM_SHA256	203,897	96	150
TLS_AES_128_CCM_SHA256	199,725	96	150

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, Client 1, Client 2 (see section A.1).

6.2 TLS Persistent Results

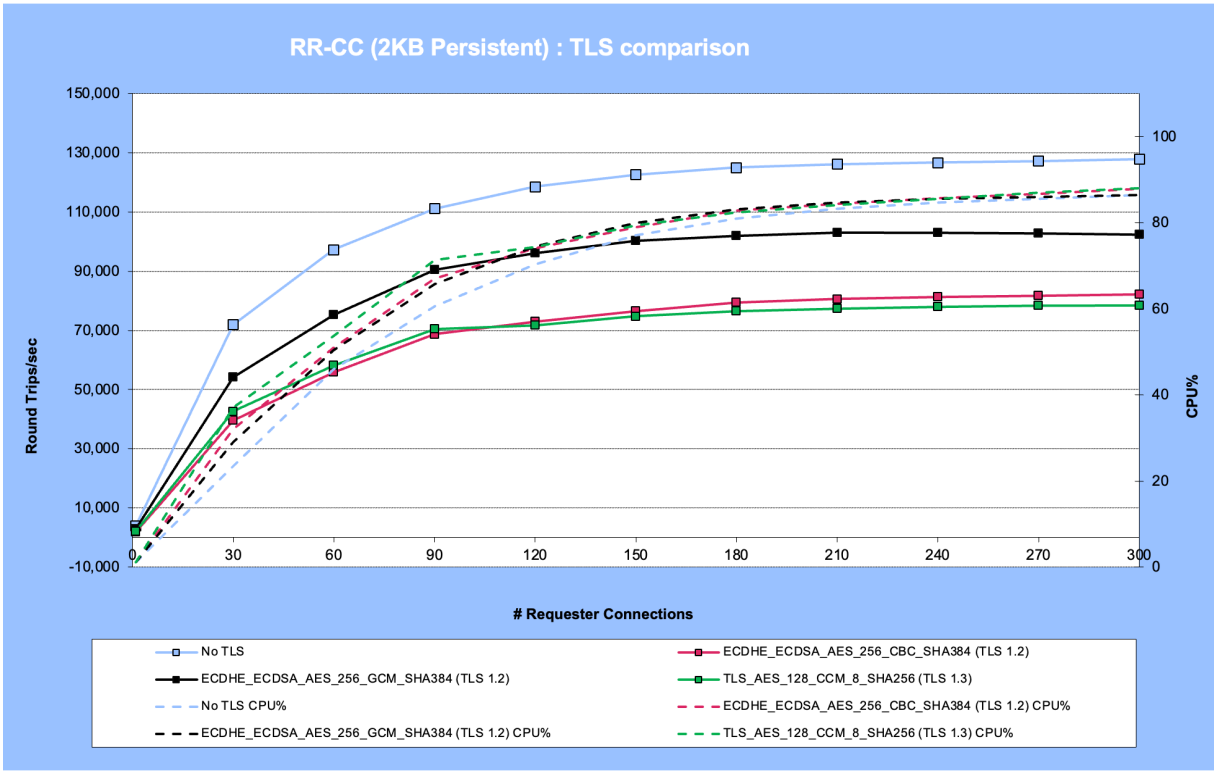


FIGURE 11 - PERFORMANCE RESULTS FOR RR-CC (2KB PERSISTENT) WITH TLS

Generally, the persistent TLS measurements showed a similar pattern to non-persistent tests (though the message rates were significantly lower throughout, as expected).
Table 11 and Table 12 show the peak throughputs for TLS 1.2 & TLS1.3 CipherSpecs.

TLS 1.2 CipherSpec	V10.0 GM Max Rate*	CPU%	Clients
No TLS	127,814	86	300
ECDHE_ECDSA_AES_256_CBC_SHA384	81,914	88	300
ECDHE_ECDSA_AES_256_GCM_SHA384	104,645	85	210
ECDHE_RSA_AES_256_CBC_SHA384	82,103	88	300
ECDHE_RSA_AES_256_GCM_SHA384	103,101	85	210

***Round trips/sec**

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

TLS 1.3 CipherSpec	V10.0 GM Max Rate*	CPU%	Clients
No TLS	127,814	86	300
TLS_AES_128_CCM_8_SHA256	78,025	88	300
TLS_AES_256_GCM_SHA384	82,675	88	300
TLS_CHACHA20_POLY1305_SHA256	78,458	88	300
TLS_AES_128_GCM_SHA256	80,029	87	270
TLS_AES_128_CCM_SHA256	80,576	88	300

***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, Client 1, Client 2 (see appendix 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, client1 & client2 are three identical machines:

- ThinkSystem SR630 V3
- 2 x 16 core CPUs. Core: Intel(R) Xeon(R) Gold 6544Y @ 3.60GHz
- 256GB RAM
- Queue manager recovery log and queue data stored locally on 2 x 1.5TB NVMe SSDs (INTEL SSDPF21Q016TB) in RAID 0 array, unless otherwise specified.
- 100Gb ethernet adapters connect all three machines via 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

- Red Hat Enterprise Linux release 9.8 (Plow)
- JMSPerfHarness test driver (see Appendix C:)
- MQ-CPH MQI test driver (see Appendix C:)
- IBM MQ V10.0

A.1.3 Software

Red Hat Enterprise Linux release 9.8 (Plow)
MQ-CPH MQI test driver (see Appendix C:)
IBM MQ V10.0

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 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

NFS mount for the MQ recovery log in NFS tests used the
following parameters:
rsize=1048576,wsize=1048576
```

A.2.2 IBM MQ

The following parameters are added or modified in the qm.ini files for the tests run in section 4,5,6 of this report:

```
Channels:
   MQIBindType=FASTPATH
   MaxActiveChannels=5000
   MaxChannels=5000
Log:
   LogBufferPages=4096
   LogFilePages=16384
   LogPrimaryFiles=16
   LogSecondaryFiles=2
   LogType=CIRCULAR
   LogWriteIntegrity=TripleWrite
SSL:
   AllowTLSV13=Yes
   MinimumRSAKeySize=2048
```

```
TuningParameters:
   DefaultPQBufferSize=10485760
   DefaultQBufferSize=10485760
```

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

```
LogFilePages=32767
LogPrimaryFiles=64
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

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