

# IBM MQ V9.3 for Linux (x86-64 platform) Performance Report

Version 1.0 - September 2022

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, February 2022.**

This edition applies to *IBM MQ V9.3* (and to all subsequent releases and modifications until otherwise indicated in new editions).

© Copyright International Business Machines Corporation 2022. 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 V9.3. 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 V9.3.

### **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 v9.3 for Linux on x86\_64.
- Want to understand the performance limits of IBM MQ v9.3 for Linux on x86\_64.
- Want to understand what actions may be taken to tune IBM MQ v9.3 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:

[https://ibm-messaging.github.io/mqperf/MQ\\_Performance\\_Best\\_Practices\\_v1.0.1.pdf](https://ibm-messaging.github.io/mqperf/MQ_Performance_Best_Practices_v1.0.1.pdf)

### Contents

This report includes:

Release highlights with performance charts.

- 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](mailto:paul.harris@uk.ibm.com).

## Contents

Preface .....	5
1 Introduction .....	8
2 Release Highlights .....	9
2.1 Streaming Queues .....	9
3 Base MQ Performance Workloads .....	11
3.1 RR-CB Workload .....	12
3.2 RR-BB Workload .....	13
3.3 RR-CC Workload .....	13
3.4 RR-DQ-BB Workload .....	14
4 Non-Persistent Performance Test Results .....	15
4.1 RR-CB Workload .....	15
4.1.1 Test setup .....	16
4.2 RR-DQ-BB Workload .....	16
4.2.1 Test setup .....	17
4.3 RR-CC JMS Workload .....	18
4.3.1 Test setup .....	19
4.4 RR-CC Workload with TLS .....	20
4.4.1 Test setup .....	21
5 Persistent Performance Test Results .....	22
5.1 RR-BB Workload .....	22
5.1.1 Test setup .....	24
5.2 Impact of Different File Systems on Persistent Messaging Performance .....	24
1.1.1 Test setup .....	25
Appendix A: Test Configurations.....	26
A.1 Hardware/Software – Set1.....	26
A.1.1 Hardware .....	26
A.1.2 Software.....	26
A.1.3 Software.....	26
A.2 Tuning Parameters Set for Measurements in This Report.....	27
A.2.1 Operating System .....	27
A.2.2 IBM MQ .....	28
Appendix B: Glossary of terms used in this report .....	29
Appendix C: Resources .....	30

## TABLES

Table 1 - Workload types.....	11
Table 2 - Peak rates for workload RR-CB (non-persistent) .....	16
Table 3 – Full Results for workload RR-DQ-BB (non-persistent) .....	17
Table 4 - Peak rates for JMS (non-persistent) .....	18
Table 5 - Peak rates for MQI client bindings (2KB non-persistent) – TLS 1.2 .....	21
Table 6 - Peak rates for MQI client bindings (2KB non-persistent) – TLS 1.3 .....	21
Table 7 - Peak rates for workload RR-BB (non-persistent) .....	23
Table 8 - Peak rates for workload RR-BB (Persistent) .....	23
Table 9 - TABLE 9 - Peak rates for workload RR-BB (Persistent SSD vs SAN vs NFS).....	25

## FIGURES

Figure 1 : Queue depth of Streaming Queue with CAPEXPY of 60 .....	10
Figure 2 - Requester-responder with remote queue manager (local responders).....	12
Figure 3 - Requester-responder with remote queue manager (remote responders).....	14
Figure 4 - Performance results for RR-CB (2KB non-persistent) .....	15
Figure 5 - Performance results for RR-DQ-BB (2KB non-persistent) .....	17
Figure 6 - Performance results for RR-CC (2KB JMS non-persistent) .....	18
Figure 7 - Performance Results for RR-CC with TLS 1.2 .....	20
Figure 8 - Performance results for RR-BB (2KB Non-persistent vs Persistent).....	22
Figure 9 - Performance Results for RR-BB Persistent Messaging logging to SSD, SAN & NFS .....	24

## 1 Introduction

IBM MQ V9.3 is a long term service (LTS) release of MQ, which includes features made available in the V9.2.1, V9.2.2, V9.2.3, V9.2.4 & V9.2.5 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 V9.2 release of IBM MQ.

Note that the tests in this report have been run on hardware significantly newer and more powerful than for the previous V9.2 report. Please bear this in mind should you compare the two reports. For example, the main MQ servers used for the V9.2 and V9.3 reports are:

### **MQ V9.2 Report**

- Lenovo System x3550 M5 – [5463-L2G]
- 2 x 12 core CPUs.
- Core: Intel® Xeon® E5-2690 v3 @ 2.60GHz
- 128GB RAM
- Queue manager recovery log and queue data stored locally 2 x 447GB SSDs (MTFDDAK480MBB) in RAID 0 array, unless otherwise specified.
- 40Gb ethernet adapters connect all three machines via an isolated performance LAN.

### **MQ V9.3 Report**

- ThinkSystem SR630 V2– [7Z71CTO1WW]
- 2 x 16 core CPUs.  
Core: Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz
- 256GB RAM
- Queue manager recovery log and queue data stored locally on 2 x 3.2TB NVMe SSDs (KCM61VUL3T20) in RAID 0 array, unless otherwise specified.
- 100Gb ethernet adapters connect all three machines via an isolated performance LAN.

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 9.3 documentation here:

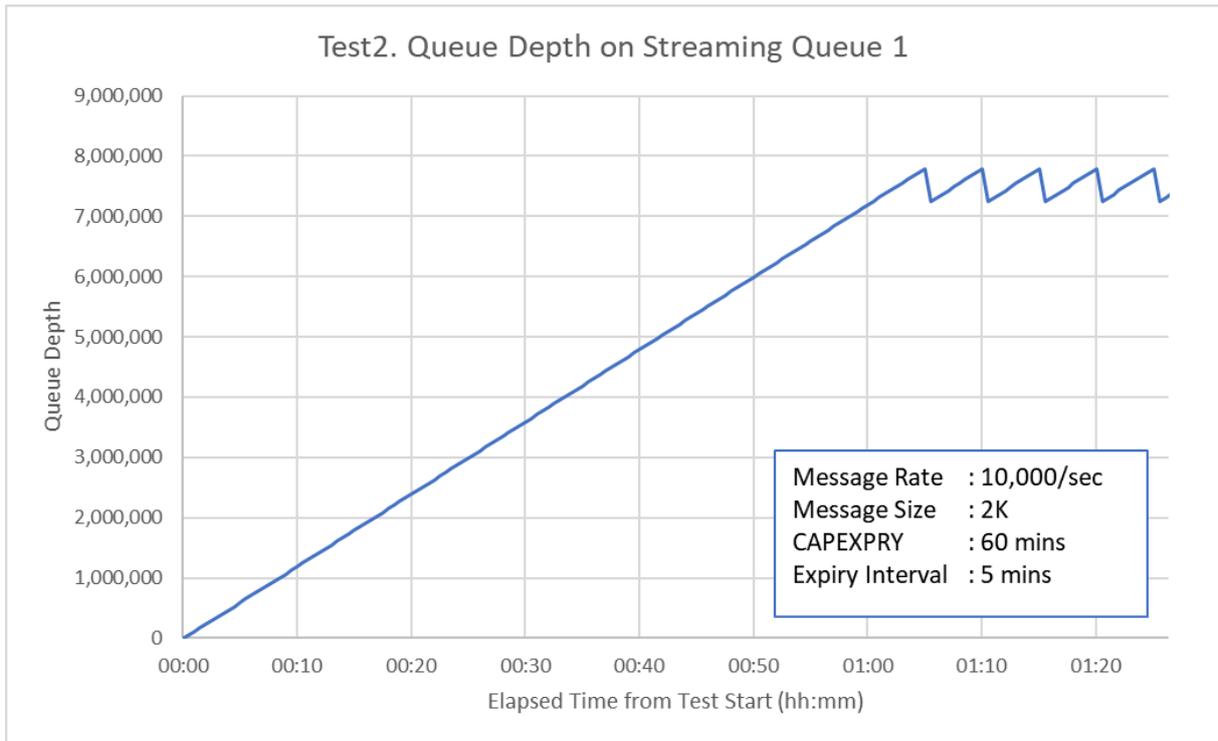
<https://www.ibm.com/docs/en/ibm-mq/9.3?topic=930-whats-new-in-mq>

Release highlights are largely functional, but a performance evaluation has been made of the new streaming queues feature.

### 2.1 Streaming Queues

Streaming queues were introduced in MQ V9.2.3 and allow you to configure a queue to put a near-identical copy of every message to a second queue. One use for streaming queues is to create duplicate messages which will be stored for a short period of time as a contingency measure. In this case the CAPEXPY custom property is set on the streaming queue to set a time limit on messages. MQ schedules an expiry task run at an interval specified in the mq.ini file (default 5 minutes). This task deletes any messages that are older than the CAPEXPY value set on it (CAPEXPY has queue scope, and only affects messages arriving after it has been set, similarly, amending the CAPEXPY value will only affect subsequent messages).

If the messages on a streaming queue with CAPEXPY set are not being consumed, then the queue will grow until the expiry task is scheduled and finds messages that are old enough to be deleted. Figure 1 below shows the queue depth of a streaming queue.



**FIGURE 1 : QUEUE DEPTH OF STREAMING QUEUE WITH CAEXPRY OF 60**

In this example, messages are arriving at a rate of 10,000/sec across 5 queues. The plot shows the depth of one of the queues, where CAEXPRY is set to 60 minutes and ExpiryInterval (the qm.ini file parameter which determines the frequency that the expiry task is scheduled) is set to 5 minutes. In this case the queue grows until the expiry task encounters messages that are 60 minutes or more old. At that point, each time the expiry task is run it will find another 5 minutes' worth of messages that can be deleted, creating the saw-tooth effect above.

When using message expiry with streaming queues you should consider the depth that the queues will grow to and the additional file system storage that may be required.

A more detailed performance report on Streaming Queues was released for V9.2.3 which can be accessed on the MQ performance GitHub page here: [MQ V9.2.3 Streaming Queues Performance Report V1.1](#)

### 3 Base MQ Performance Workloads

Table 1 (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).

<b>Workload</b>	<b>Description</b>
<b>RR-CB</b>	Client mode requesters on separate host. Binding mode responders.
<b>RR-DQ-BB</b>	Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders.
<b>RR-BB</b>	Binding mode requesters and responders
<b>RR-CC</b>	Client mode requesters, and responders on separate, unique hosts

**TABLE 1 - 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.

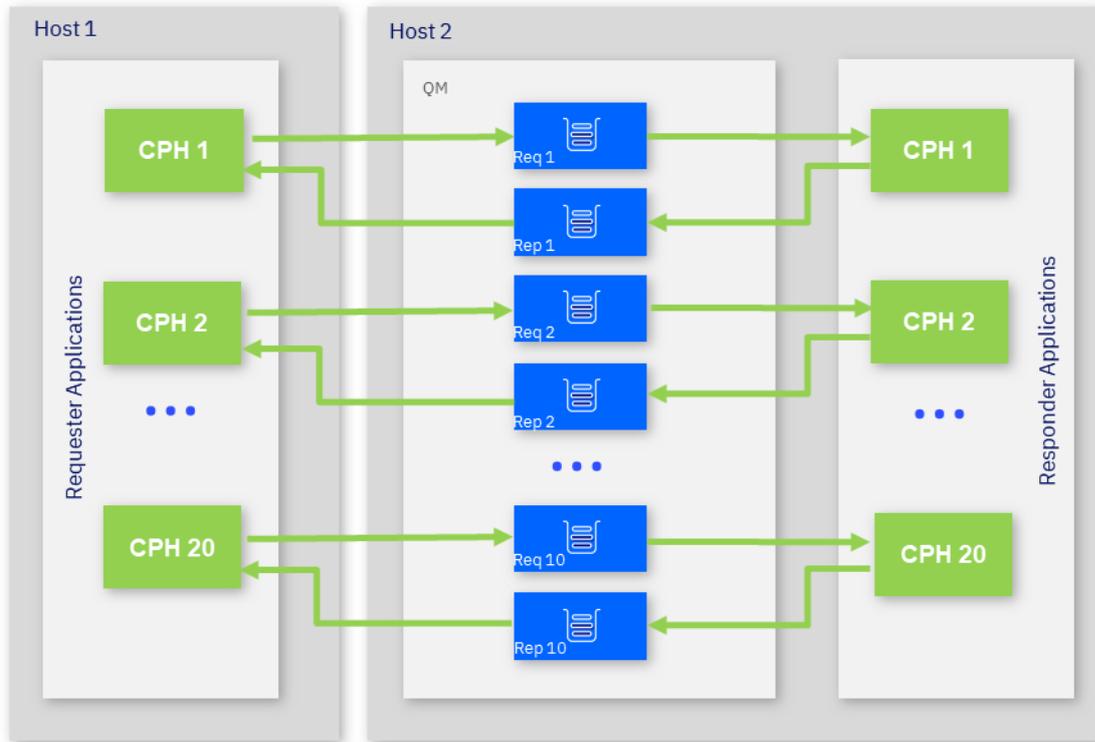
RR-CB & RR-DQ-BB are described in the following section. The remaining two workloads differ only in the location of the MQ applications, which is made clear in the results presented in this report.

#### [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-CB Workload

(Client mode requesters on separate host. Binding mode responders.)



**FIGURE 2 - REQUESTER-RESPONDER WITH REMOTE QUEUE MANAGER (LOCAL RESPONDERS)**

Figure 2 shows the topology of the RR-CB 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.

### 3.2 RR-BB Workload

(Binding mode requesters with binding mode responders.)

This workload is run the same way as RR-CB above, but with the requesters running in binding mode, on the same host as the queue manager.

### 3.3 RR-CC Workload

(Client mode requesters with client mode responders.)

This workload is run the same way as RR-CB above, but with the responders running in client mode, on a dedicated, remote host.



## 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-CB Workload

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

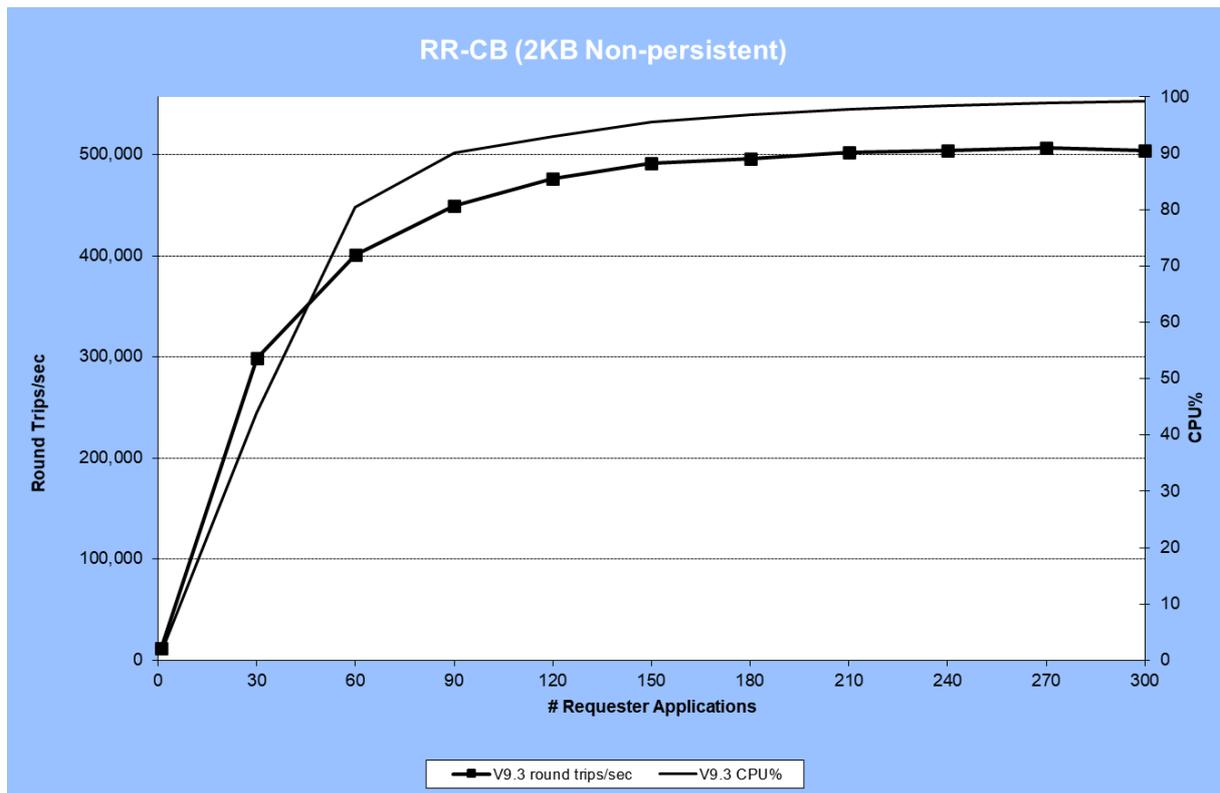


FIGURE 4 - PERFORMANCE RESULTS FOR RR-CB (2KB NON-PERSISTENT)

The test peaked at approximately 500,000 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.

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-CB (2KB Non-persistent)	506,887	98.88	270
RR-CB (20KB Non-persistent)	332,597	95.29	150
RR-CB (200KB Non-persistent)	52,659	43.03	100
RR-CB (2MB Non-persistent)	5,064	35.87	45

**\*Round trips/sec**

**TABLE 2 - PEAK RATES FOR WORKLOAD RR-CB (NON-PERSISTENT)**

#### 4.1.1 Test setup

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

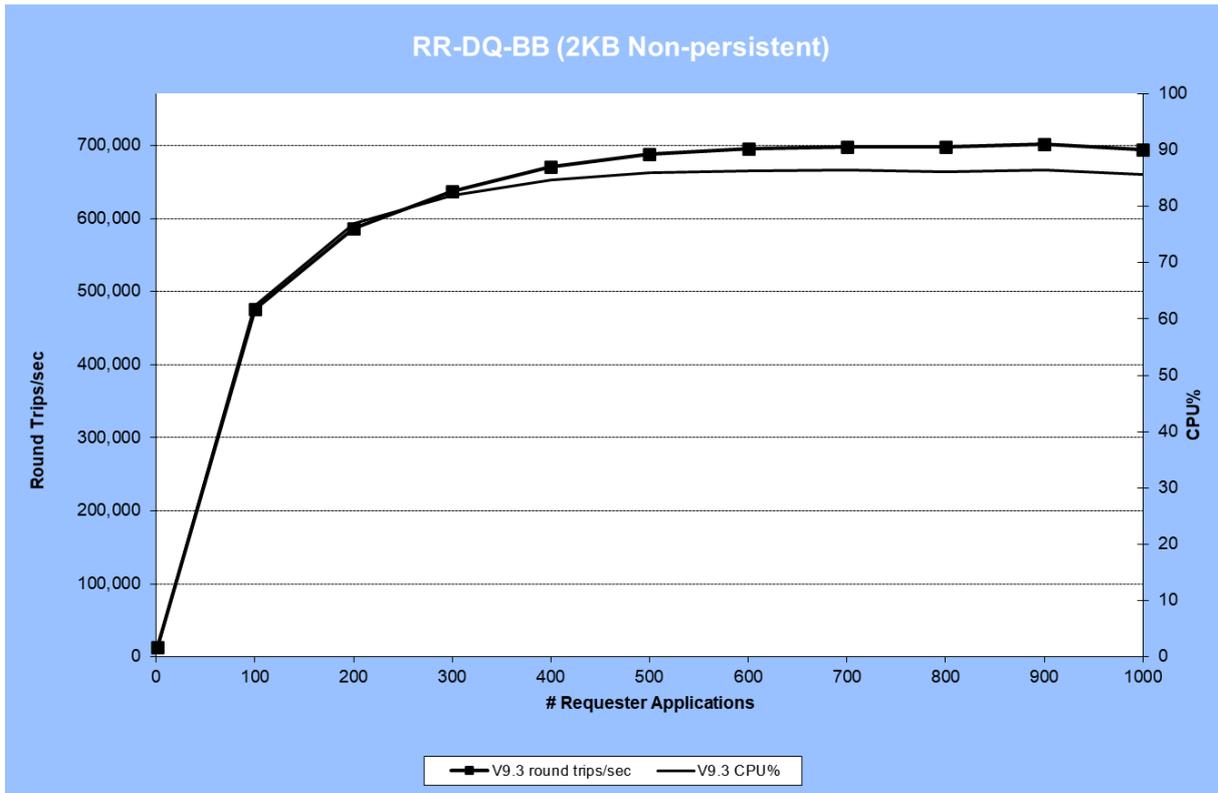
Hardware: Server 1, Client 1, Client 2 (see section A.1).

## 4.2 RR-DQ-BB Workload

(Distributed queuing 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 3.4) 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 5 - 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.

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-DQ-BB (2KB Non-persistent)	701,776	86.42	900
RR-DQ-BB (20KB Non-persistent)	350,161	61.47	240
RR-DQ-BB (200KB Non-persistent)	57,076	29.42	50
RR-DQ-BB (2MB Non-persistent)	5,497	38.45	40

**\*Round trips/sec**

**TABLE 3 – FULL RESULTS FOR WORKLOAD RR-DQ-BB (NON-PERSISTENT)**

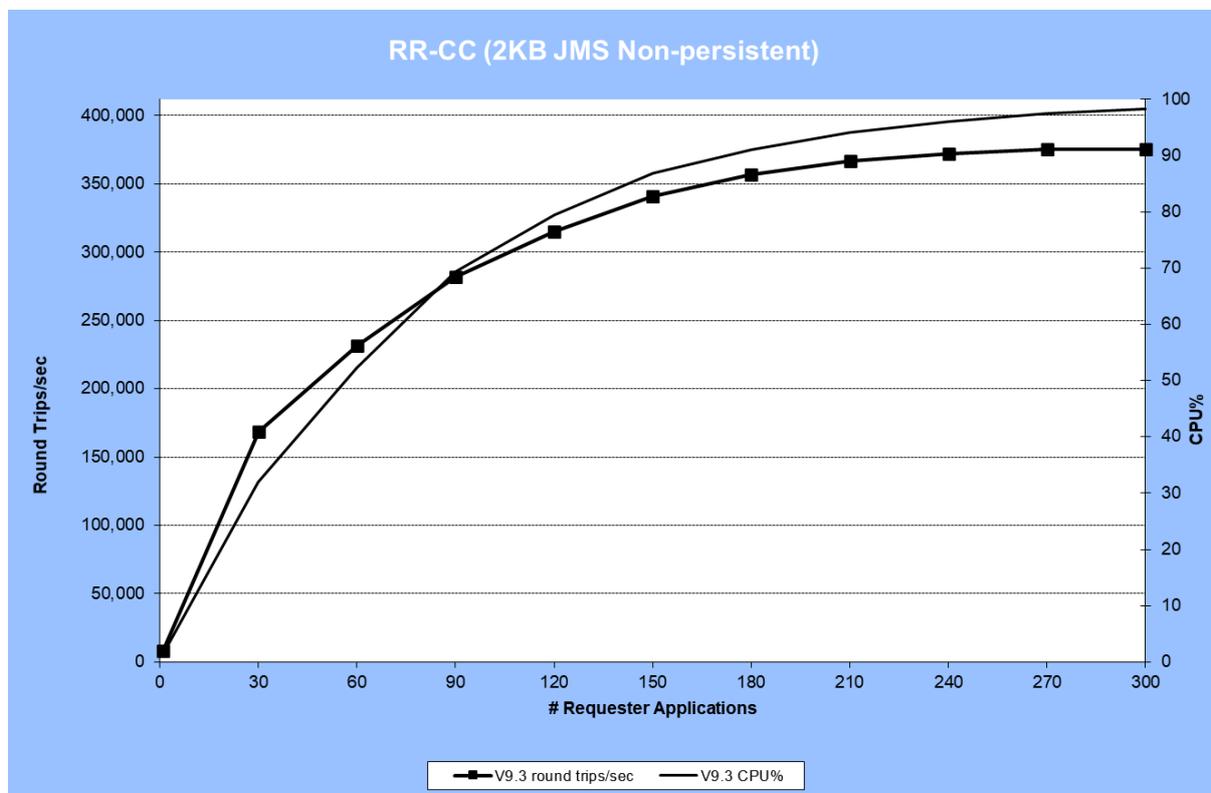
#### 4.2.1 Test setup

Workload type: RR-DQ-BB (see section 3.4).

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 6 - 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 375,000 round trips/sec

Peak round trip rates for all message sizes tested can be seen in the table below. The 200KB and 2MB scenarios are network limited by the 100Gb network; the rates are lower than the RR-CB network limited scenarios because of the additional network hop to the responder applications which are local in the RR-CB scenario.

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-CC (2KB JMS Non-persistent)	375,453	98.29	300
RR-CC (20KB JMS Non-persistent)	252,003	95.57	240
RR-CC (200KB JMS Non-persistent)	27,580	55.44	200
RR-CC (2MB JMS Non-persistent)	2,653	51.66	200

\*Round trips/sec

**TABLE 4 - PEAK RATES FOR JMS (NON-PERSISTENT)**

#### 4.3.1 Test setup

Workload type: RR-CC (see section 3.3).

Message protocol: JMS

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

## 4.4 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 6 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).

- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- 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.

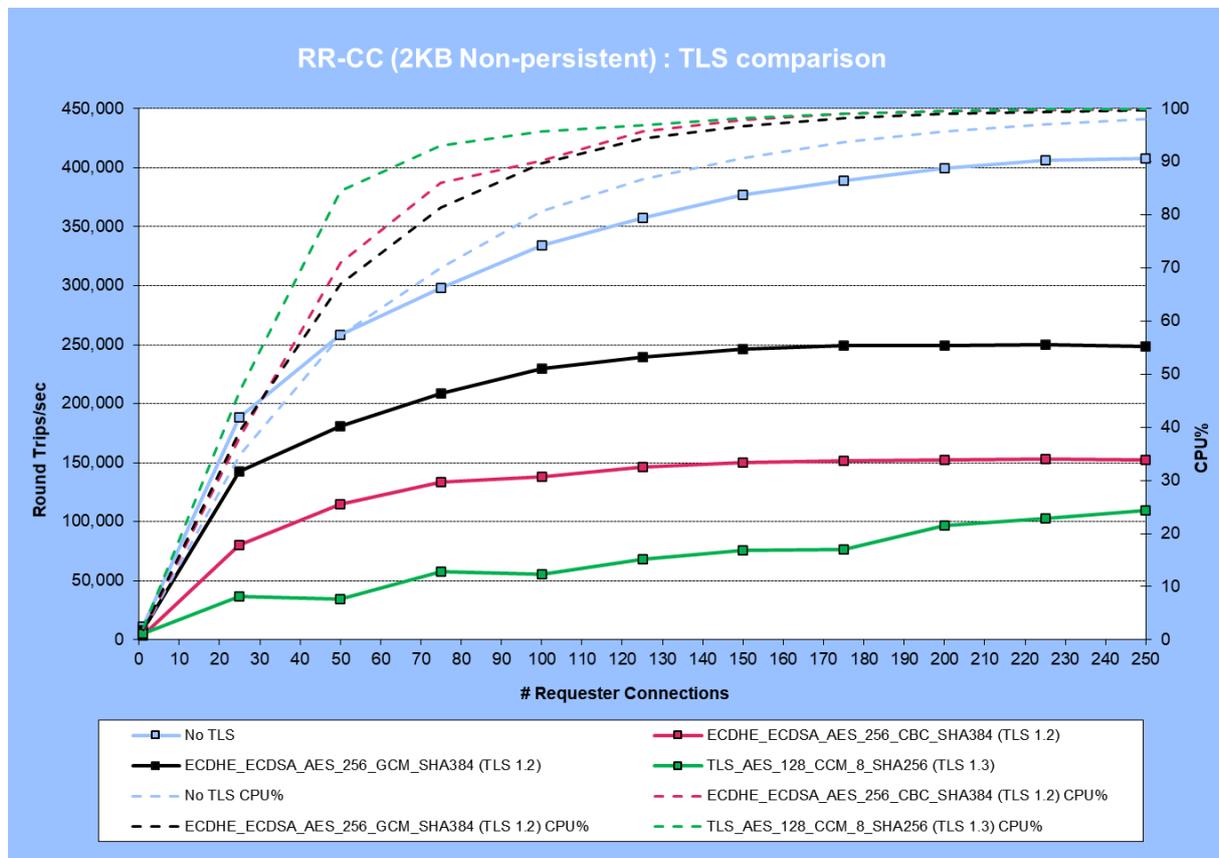


FIGURE 7 - PERFORMANCE RESULTS FOR RR-CC WITH TLS 1.2

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 of 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 62% of the throughput of a non-encrypted workload. CBC based CipherSpecs exhibited a greater overhead, running at approximately 37% of a non-encrypted workload. TLS 1.3 encryption is more expensive, achieving rates slightly below the TLS 1.2 CBC based CipherSpecs.

Table 5 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	V9.3 GM		
	Max Rate*	CPU%	Clients
No TLS	407,879	98	250
TLS_RSA_WITH_AES_256_CBC_SHA256	172,304	100	225
TLS_RSA_WITH_AES_256_GCM_SHA384	253,979	99	200
ECDHE_ECDSA_AES_256_CBC_SHA384	152,771	100	225
ECDHE_ECDSA_AES_256_GCM_SHA384	250,062	99	225
ECDHE_RSA_AES_256_CBC_SHA384	152,203	100	225
ECDHE_RSA_AES_256_GCM_SHA384	252,200	99	225

*\*Round trips/sec*

**TABLE 5 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB NON-PERSISTENT) – TLS 1.2**

Table 6 shows the peak rates achieved for all TLS 1.3 CipherSpecs.

TLS 1.3 CipherSpec	V9.3 GM		
	Max Rate*	CPU%	Clients
No TLS	407,879	98	250
TLS_AES_128_CCM_8_SHA256	109,306	100	250
TLS_AES_256_GCM_SHA384	107,994	100	250
TLS_CHACHA20_POLY1305_SHA256	107,995	100	250
TLS_AES_128_GCM_SHA256	106,100	100	250
TLS_AES_128_CCM_SHA256	110,713	100	250

*\*Round trips/sec*

**TABLE 6 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB NON-PERSISTENT) – TLS 1.3**

#### 4.4.1 Test setup

Workload type: RR-CC (see section 3.3).

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 dependant 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 V9.3 in a separate environment (x64 Linux with SAN, SSD & NFS filesystems) are shown to demonstrate the impact of recovery log location.

### 5.1 RR-BB Workload

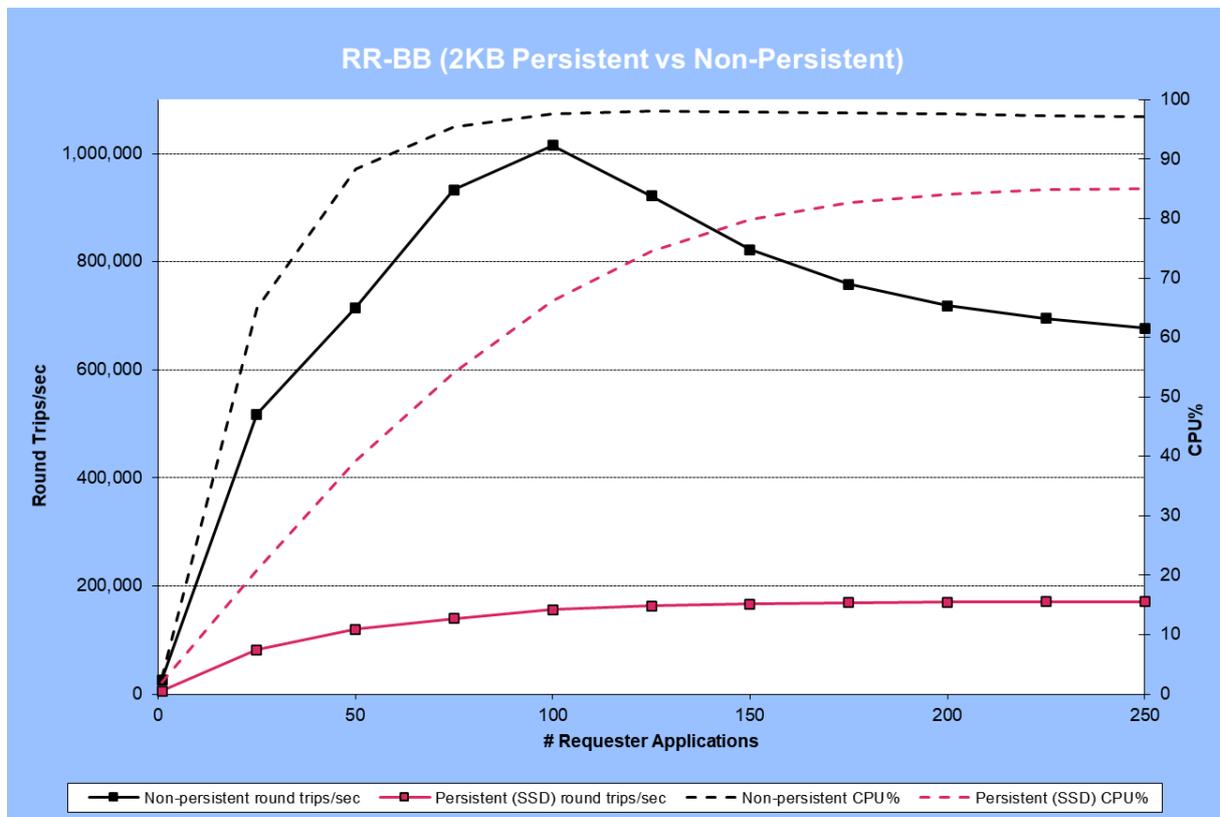


FIGURE 8 - PERFORMANCE RESULTS FOR RR-BB (2KB NON-PERSISTENT VS PERSISTENT)

RR-BB (see section 3.2) is a variant of RR-CB where all applications are connected in bindings mode. This accentuates the impact of persistent messaging since we are no longer limited by network bandwidth.

Figure 8 shows results from running the RR-BB 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 100 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).

Peak round trip rates for all message sizes tested, for persistent & non-persistent scenarios can be seen in Table 7 & Table 8 below.

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-BB (2K Non-persistent)	1,034,401	97.46	96
RR-BB (20K Non-persistent)	244,023	32.15	12
RR-BB (200K Non-persistent)	133,707	53.33	28
RR-BB (2MB Non-persistent)	7,296	44.86	28

*\*ROUND TRIPS/ SEC*

**TABLE 7 - PEAK RATES FOR WORKLOAD RR-BB (NON-PERSISTENT)**

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-BB (2KB Persistent)	171,103	84.96	250
RR-BB (20KB Persistent)	109,803	60.53	175
RR-BB (200KB Persistent)	15,030	20.93	70
RR-BB (2MB Persistent)	1,579	14.24	16

*\*ROUND TRIPS/ SEC*

**TABLE 8 - PEAK RATES FOR WORKLOAD RR-BB (PERSISTENT)**

With the larger machines used for this report there are limits to these non-persistent workloads before CPU utilisation is the bottleneck. At the highest data transfer rate (200K non-persistent messaging in this case) MQ is transferring data at around 55 GBytes/sec however. Additional queue managers may help though having all applications running locally is not typically representative of the real world. 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-BB (see section 3.2).

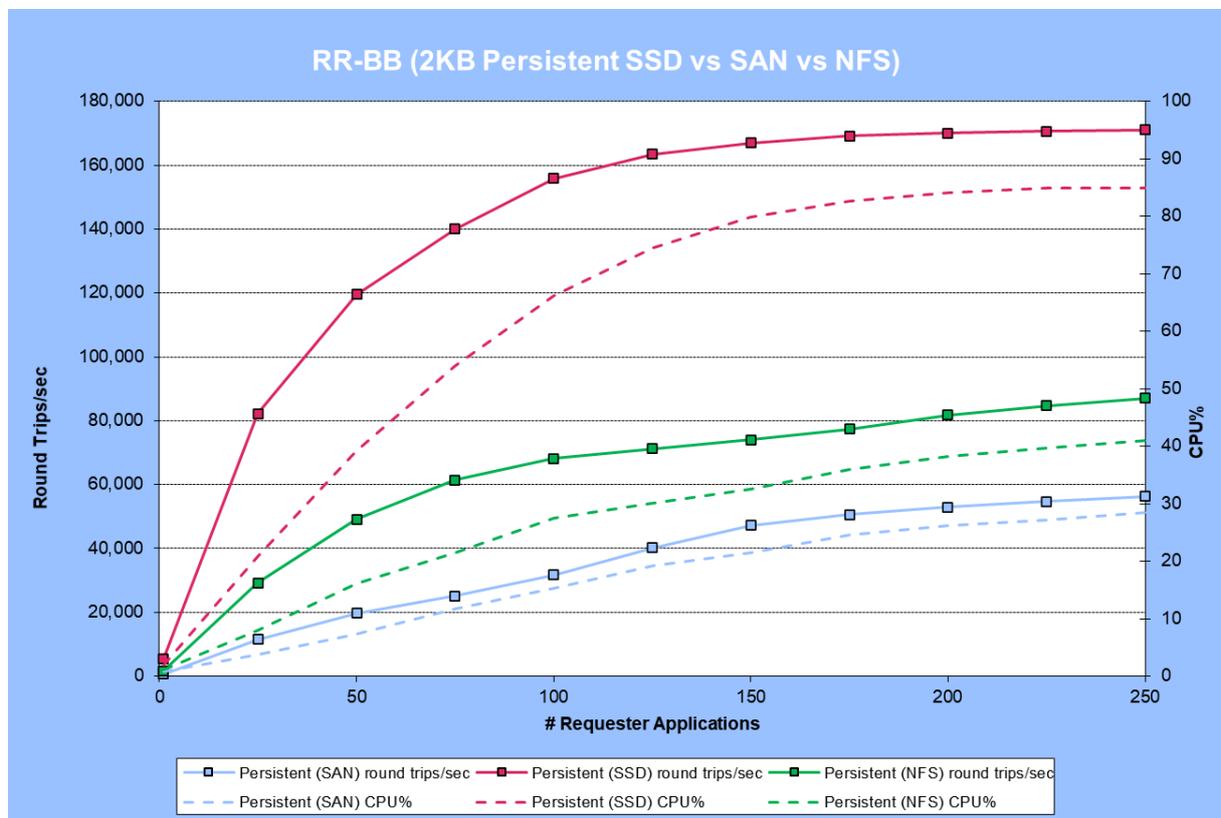
Hardware: Server 1 (see section A.1).

## 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](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-BB 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-BB workload with persistent messaging where the recovery logs are on a local SSD or hosted remotely (SAN or NFS).

RR-BB (see section 3.2) is a variant of RR-CB where all applications are connected in bindings mode, eliminating the network traffic (except in the case of NFS, where there is a 100Gb link from the MQ server to the NFS server).

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 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 for NFS, in the Linux kernel.

Table 9 below, shows the peak rates achieved for each filesystem tested, across a range of message sizes.

Test	V9.3		
	Max Rate*	CPU%	Clients
RR-BB (2KB Persistent - SSD)	171,103	84.96	250
RR-BB (2KB Persistent - SAN)	56,295	28.46	250
RR-BB (2KB Persistent - NFS)	87,082	41.02	250
RR-BB (20KB Persistent - SSD)	109,803	60.53	175
RR-BB (20KB Persistent - SAN)	18,251	13.67	250
RR-BB (20KB Persistent - NFS)	21,894	13.99	250
RR-BB (200KB Persistent - SSD)	15,030	20.93	70
RR-BB (200KB Persistent - SAN)	2,727	5.86	70
RR-BB (200KB Persistent - NFS)	3,048	5.65	70
RR-BB (2MB Persistent - SSD)	1,579	14.24	16
RR-BB (2MB Persistent - SAN)	300	4.11	24
RR-BB (2MB Persistent - NFS)	337	3.76	20

*\*Round trips/ sec*

**TABLE 9 - TABLE 9 - PEAK RATES FOR WORKLOAD RR-BB (PERSISTENT SSD vs SAN vs NFS)**

### 1.1.1 Test setup

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

Hardware: Server 1, with client 1 machine acting as NFS server. (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, client1 & client2 are three identical machines:

- ThinkSystem SR630 V2– [7Z71CTO1WW]
- 2 x 16 core CPUs.  
Core: Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz
- 256GB RAM
- Queue manager recovery log and queue data stored locally on 2 x 3.2TB NVMe SSDs (KCM61VUL3T20) 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.

SAN Infrastructure:

- IBM 2498-F48 fibre channel SAN switch (16Gb/s ports)
- IBM SAN Volume Controller (2145-SV1) with 256GB RAM
- IBM Flash System 900 Storage.

#### A.1.2 Software

- Red Hat Enterprise Linux Server release 8.5 (Ootpa)
- JMSPerfHarness test driver (see Appendix C:)
- MQ-CPH MQI test driver (see Appendix C:)
- IBM MQ V9.3

#### A.1.3 Software

Red Hat Enterprise Linux Server release 7.7 (Maipo)

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

IBM MQ V9.3

## 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
kernel.sched_latency_ns = 2000000
kernel.sched_min_granularity_ns = 1000000
kernel.sched_wakeup_granularity_ns = 400000
```

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

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

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

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 of this report:

### Channels:

```
MQIBindType=FASTPATH
MaxActiveChannels=5000
MaxChannels=5000
```

### Log:

```
LogBufferPages=4096
LogFilePages=16384
LogPrimaryFiles=16
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 ( <a href="https://github.com/ot4i/perf-harness">https://github.com/ot4i/perf-harness</a> )
LTS	Long term service.
MQ-CPH	C based, performance test application ( <a href="https://github.com/ibm-messaging/mq-cph">https://github.com/ibm-messaging/mq-cph</a> )

## Appendix C: Resources

MQ Performance GitHub Site

<https://ibm-messaging.github.io/mqperf/>

Streaming Queues Performance Paper

[MQ V9.2.3 Streaming Queues Performance Report V1.1](#)

IBM MQ Performance: Best Practises, and Tuning Paper:

[https://ibm-messaging.github.io/mqperf/MQ\\_Performance\\_Best\\_Practices\\_v1.0.1.pdf](https://ibm-messaging.github.io/mqperf/MQ_Performance_Best_Practices_v1.0.1.pdf)

MQ-CPH (The IBM MQ C Performance Harness)

<https://github.com/ibm-messaging/mq-cph>

JMSPerfHarness

<https://github.com/ot4i/perf-harness>