IBM MQ V9.1 for Linux (x86-64 platform) Performance Report

Version 1.0 - September 2018

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

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, August 2018.

This edition applies to *IBM MQ V9.1* (and to all subsequent releases and modifications until otherwise indicated in new editions).

© Copyright International Business Machines Corporation 2018. 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 V9.1. 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 V9.1.

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 v9.1 for Linux on x86_64.
- Want to understand the performance limits of IBM MQ v9.1 for Linux on x86_64.
- Want to understand what actions may be taken to tune IBM MQ v9.1 for Linux on x86_64.

The reader should have a general awareness of the Linux operating system and of IBM MQ in order 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. You may refer to the V8 performance reports, which include chapters discussing these aspects of performance. It is planned to update those sections as a stand-alone, platform neutral paper in the future. This will be published on the MQ performance GitHub page, along with other performance papers already available there:

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

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.

Contents

TABLES

FIGURES

1 Introduction

IBM MQ V9.1 is a long term service (LTS) release of MQ, which includes features made available in the V9.0.1, V9.0.2, V9.0.3, V9.0.4 & V9.0.5 continuous delivery (CD) releases. Those CD releases are mainly functional in nature, but there are some significant improvements to performance as well.

- Implicit syncpoints (see section 1.1)
- Automatic linear log management (see section 1.2)

Other areas that are significant to performance are:

- Improved guidance on transaction log sizing

  The V9.1 knowledge centre provides much more detail on how you can size you MQ transaction log. This can be seen in the subsections of the following knowledge centre item:
  https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.con.doc/q018470_.htm

- System topics for monitoring enhanced with additional metrics.

  System topics for monitoring was introduced in V9.0, and has been enhanced in the V9.0.x CD releases of the product, and now available in V9.1.
  The following new metrics can be subscribed to, or viewed with the sample program (amqsrua)

  Class(DISK), Type(Log):

  Log - write size
  Log - current primary space in use
  Log - workload primary space utilization

  Write size reports the average size of a write to the MQ transaction log, which, in combination with the latency (also reported), can give more insight to the performance of you transaction logging.

  'Current primary space in use', and 'workload primary space utilization', report the values for the LOGINUSE, and LOGUTIL attributes of DISPLAY QMSTATUS, respectively:

  https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q086250_.htm

  For more information, see the following blog articles and knowledge centre section.

  Blog Article:
  https://www.ibm.com/developerworks/community/blogs/messaging/entry/Statistics_published_to_the_system_topic_in_MQ_v9?lang=en_us

  Knowledge centre:
  https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.mon.doc/mo00040_.htm

Performance data presented in this report does not include release to release comparisons, but all tests run showed equal or better performance than V8.0 & V9.0 releases of IBM MQ.

1.1 Linear Logging Enhancements

MQ V9.1 includes significant improvements to linear logging over MQ V9.0.

- Automatic management of log extents
- Automatic recording of media images
- Increased performance to near circular log performance

See the blog article on developer works for more details:
https://www.ibm.com/developerworks/community/blogs/messaging/entry/Logger_enhancements_for_MQ_v9_0_2?lang=en

With automatic log extent re-use, the performance of linear logging approaches that of circular logging.

These features were performance tested in the V9.0.2 CD release where they were introduced.

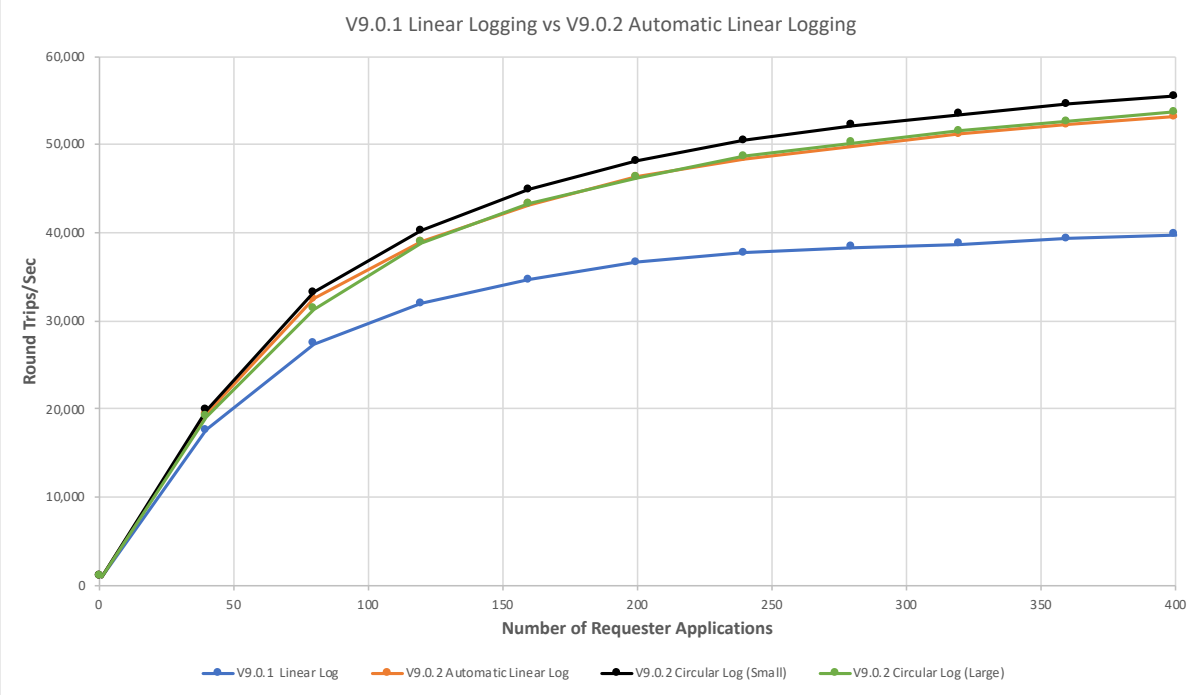

FIGURE 1 - LINEAR LOGGING PERFORMANCE

Figure 1, shows the improvement in linear logging performance, between V9.0.1 (equivalent to V9.0), and V9.0.2 (with automatic log management). A workload was run that was limited by the rate the queue manager could write to the log files (on SAN storage). The overall round trip rate is shown, which increases as we add more applications putting and getting to the queues. For both linear logging tests, media images were recorded every 10 seconds with rcdmqimg. Whilst the V9.0.1 test will continue to create and format new linear log files continuously, in V9.0.2, existing log files are made available for re-use as the media recovery point is moved forward in the log. Performance is significantly improved in V9.0.2 as a result, coming close the that of circular logging (with an efficiently sized log). Running the same test with an over-sized circular log file set (labelled 'V9.0.2 Circular Log(Large)') shows a similar level of performance to linear logging, indicating that the number of files being used is the main difference between circular and linear logging now, rather than the additional formatting costs.

1.2 Implicit Syncpoints

A long term performance recommendation has been to ensure that any MQPUT is executed within a syncpoint. This ensures that the queue being updated is not locked for the duration of the update operation, which includes a synchronous write to the transaction log. Ensuring that all PUTs are inside a syncpoint benefits workloads where there is a contention for the queue. It is also necessary for an MQGET, though getting a transactional, persistent message outside of syncpoint does not make sense from a business logic point of view either (if the message is lost at the network layer, MQ will already have destroyed it).

It is still good practise to put messages inside a syncpoint, but if an application has not been written to do this, it can sometimes be difficult to change retrospectively.

Implicit syncpoints, in MQ V9.1 (introduced in V9.0.5) detects PUTs outside of a syncpoint and 'wraps' them in a unit of work, enabling the queue manager to process the PUT more efficiently. Please refer to the following blog article for more detail: https://developer.ibm.com/messaging/2018/04/24/implicit-syncpointing-persistent-messages-put-outside-syncpoint

The figures below illustrate the benefit that can be obtained with implicit syncpoints. All data has been collected on a Linux machine.

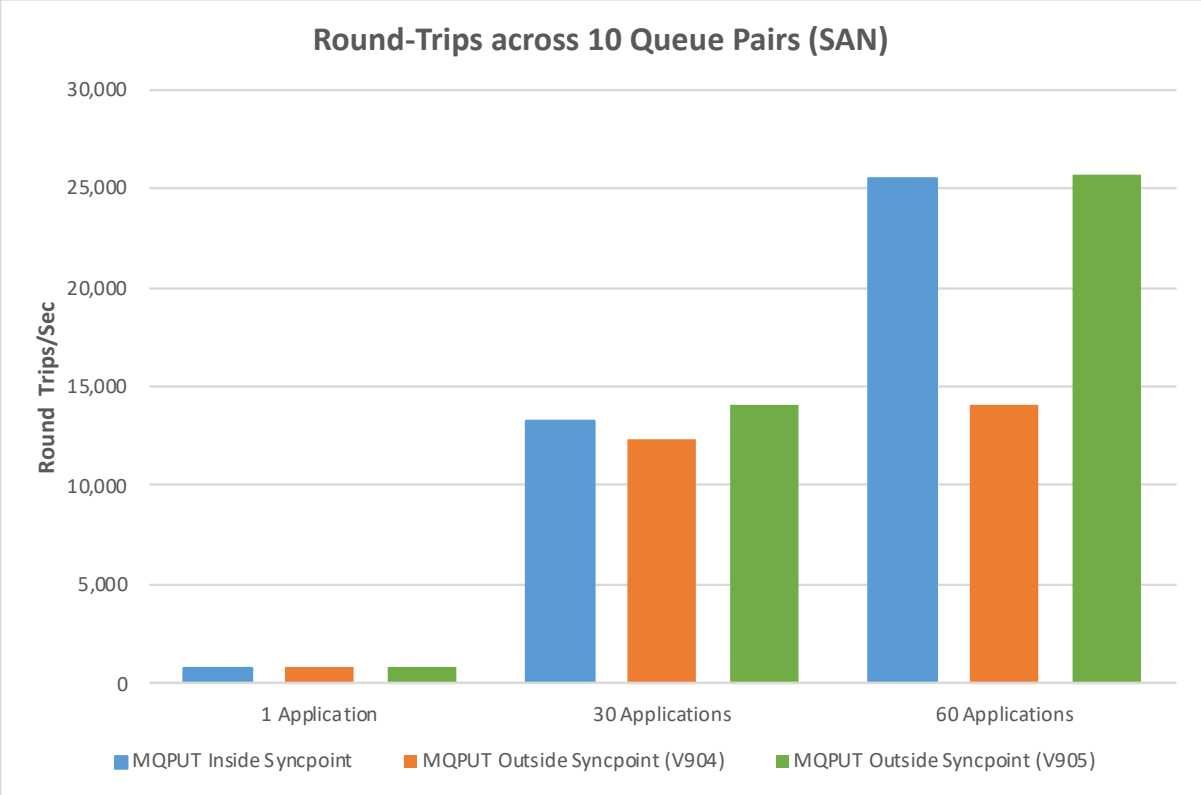

FIGURE 2 - EFFECT OF SYNCPOINT BY NUMBER OF APPLICATIONS

Figure 2 shows performance results for tests where 10 queues pairs are utilised, with an increasing number of requester applications running, processing 2KiB messages. When there is only one application, there will never be another MQPUT being processed alongside that of application 1, so there is little difference between executing the MQPUT inside, or outside of syncpoint, in the application. Once we add more MQ applications, the benefits of using syncpoints become evident, particularly with a higher latency filesystem, as MQPUTs outside of syncpoint will lock the queue while the log record is

synchronously forced to disk. Using syncpoints reduces contention with the added benefit that other applications can write into the log buffer, resulting in more aggregation of log data, in a single write. With V9.0.5, implicit syncpointing reduces lock contention, matching the performance of the explicit syncpoint scenario.

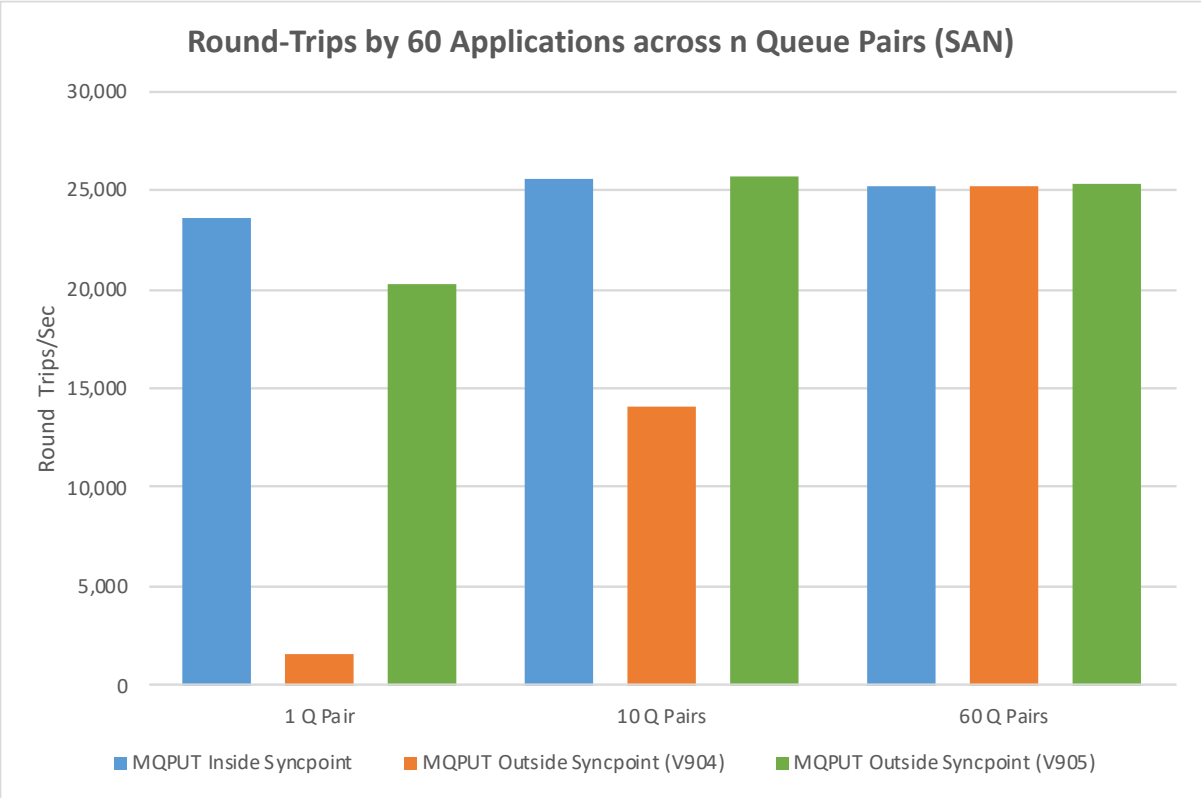

FIGURE 3 – EFFECT OF SYNCPOINT BY NUMBER OF QUEUES

Figure 3 shows the effect of reducing queue locking by spreading the load across a number of queue pairs (REQUEST Q/REPLY Q). All tests use 60 requester applications. When the workload is driven through a single pair of queues, the non-syncpoint case has a low throughput (not much better than the test using 1 requester in chart 1), as each MQPUT queues up behind the previous one to that queue, with a forced log write being executed within the scope of the queue lock. Using syncpoints alleviates this issue, allowing for more concurrency. As we increase the number of queue pairs, the locking becomes less of an issue, until, at 60 pairs of queues, where there are only 2 requester applications per queue pair, the non-syncpoint case is not much less than using syncpoints. Once again, the V.9.0.5 test case, with PUTs outside of syncpoints matches the explicit syncpoint scenario.

2 Workloads

Table 1 (below) lists the workloads used in the generation of performance data for 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-CB	Client mode requesters on separate host. Binding mode responders.
RR-DQ-BB	Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders.
RR-BB	Binding mode requesters and responders
RR-CC	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).

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.

1.2 RR-CB Workload (Client mode requesters on separate host. Binding mode responders.)

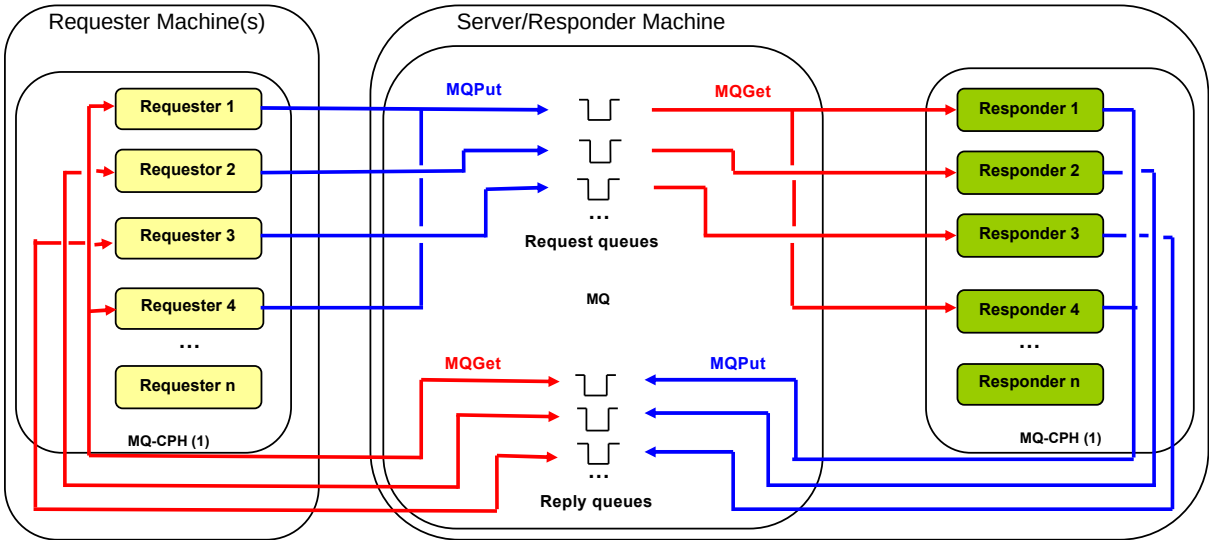

FIGURE 4 - REQUESTER-RESPONDER WITH REMOTE QUEUE MANAGER (LOCAL RESPONDERS)

Figure 4 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. Each requester is a thread running in an MQI (CPH) or JMS (JMSPerfHarness) application. Additional

machines may be used to drive the requester applications where necessary. The threads utilise the requester queues in a round robin fashion, ensuring even distribution of traffic.

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. Each responder is a thread of CPH or JMSPerfHarness and there may be multiple instances of these MQI or JMS applications, similar to the responders. The number of responders is set such that there is always a waiting 'getter' for the request queue.

The flow of the test is as follows:

1. 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.
2. 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.
3. 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 programs 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

Variants of the RR-CB test differ in the location of the applications (RR-BB & RR-CC).

1.3 RR-DQ-BB Workload (Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders).

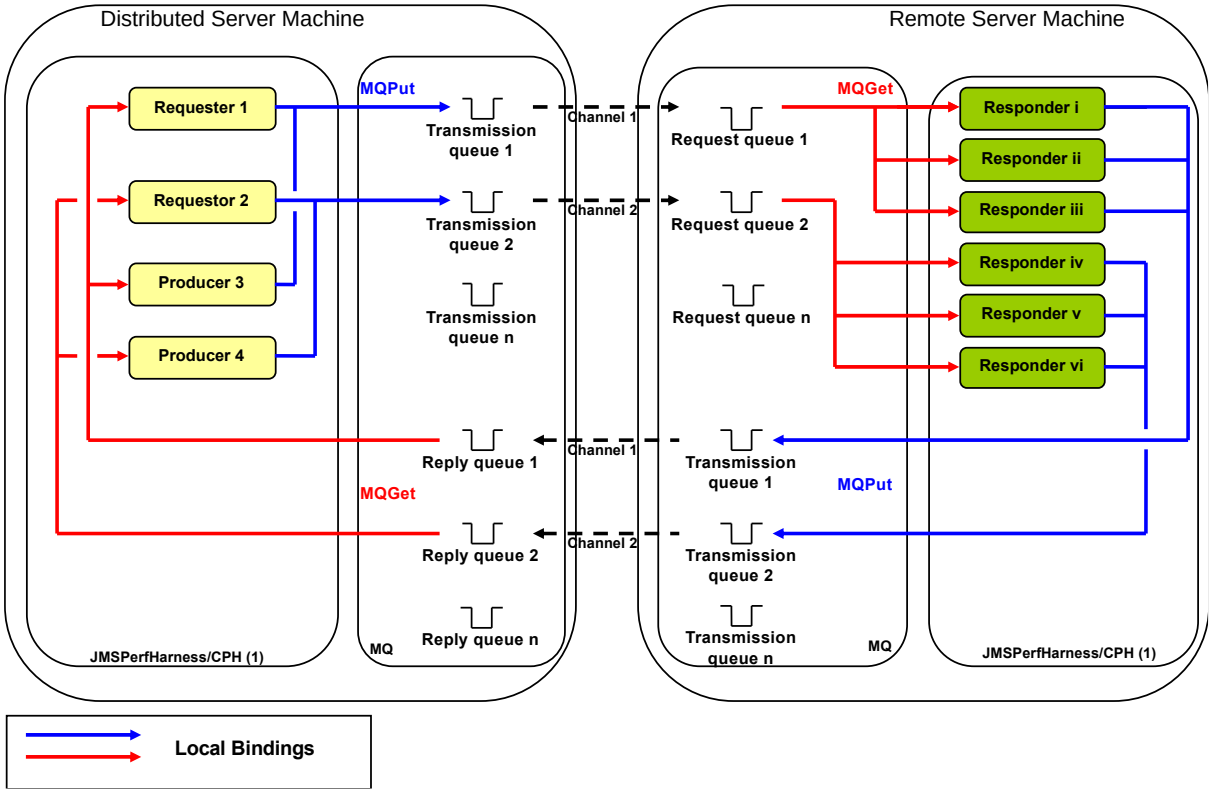

FIGURE 5 - REQUESTER-RESPONDER WITH REMOTE QUEUE MANAGER (REMOTE RESPONDERS).

This is a distributed queuing version of the requester-responder topology detailed in section 1.2. All *MQPUT*s are to remote queues so that messages are now transported across server channels to the queue manager where the queue is hosted.

2 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).

2.1 RR-CB Workload

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

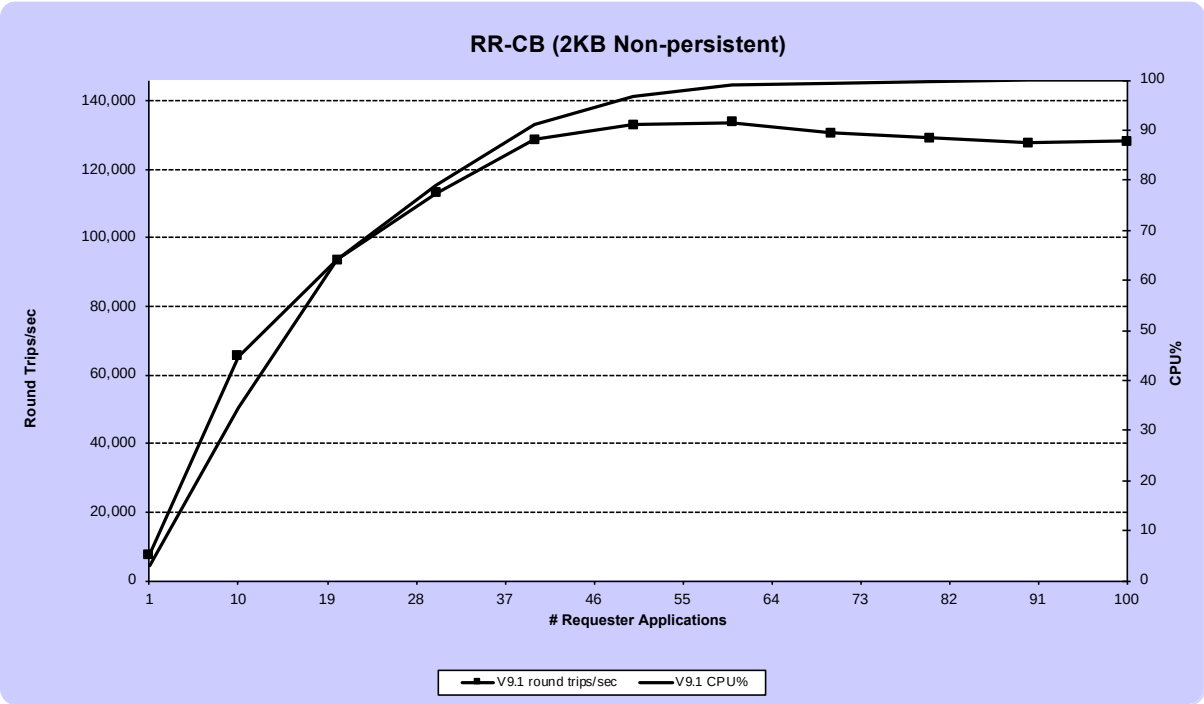

FIGURE 6 - PERFORMANCE RESULTS FOR RR-CB (2KB NON-PERSISTENT)

The test peaked at approximately 134,000 round trips/sec, fully utilising the CPU 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 being limited by the 40Gb network that links the client and server machines.

Test	V9.1 Max Rate*	CPU%	Clients
RR-CB (2KB Non-persistent)	133,588	98.93	60
RR-CB (20KB Non-persistent)	110,517	98.85	60
RR-CB (200KB Non-persistent)	22,624	59.96	50
RR-CB (2MB Non-persistent)	2,220	61.27	50

***ROUND TRIPS/ SEC**

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

2.1.1 Test setup

Workload type: RR-CB (see section 1.2).

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

2.2 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 of the tests in this section utilise multiple send/receive channels. This particularly helps with smaller, non-persistent messages when the network is under-utilised.

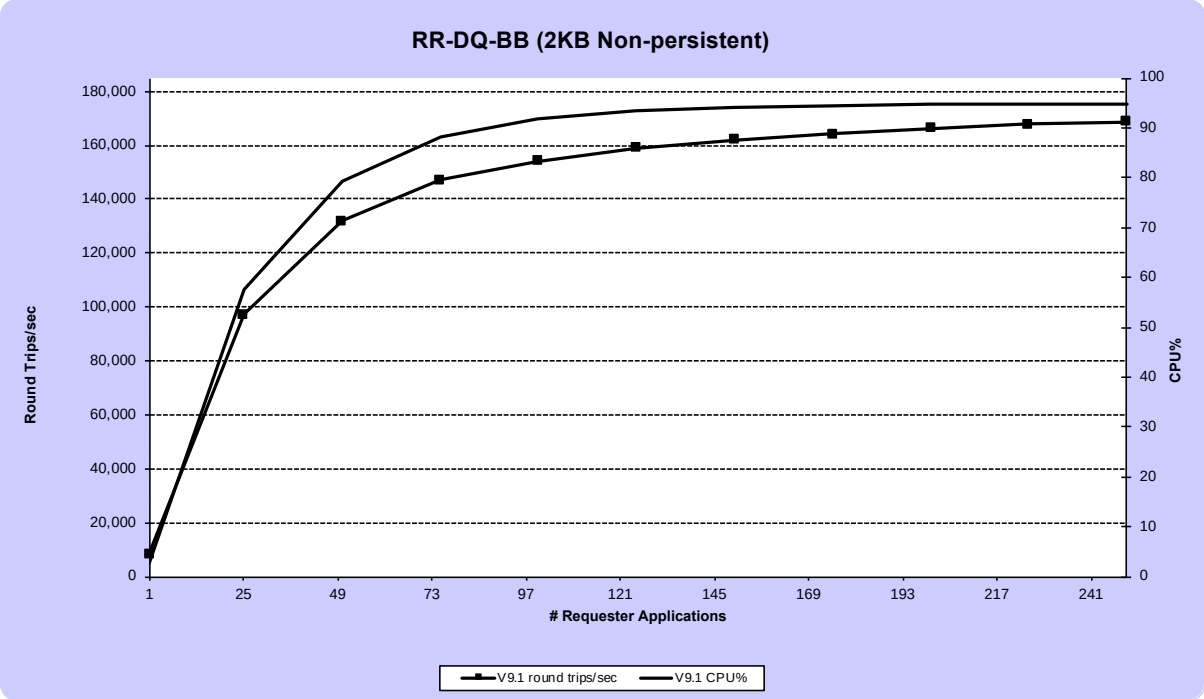

FIGURE 7 - PERFORMANCE RESULTS FOR RR-DQ-BB (2KB NON-PERSISTENT)

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 40Gb network.

Test	V9.1 Max Rate*	CPU%	Clients
RR-DQ-BB (2KB Non-persistent)	168,742	94.74	250
RR-DQ-BB (20KB Non-persistent)	124,653	92.08	150
RR-DQ-BB (200KB Non-persistent)	22,322	54.39	50
RR-DQ-BB (2MB Non-persistent)	2,122	54.29	30

***ROUND TRIPS/ SEC**

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

2.2.1 Test setup

Workload type: RR-DQ-BB (see section 0).

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

2.3 RR-CC JMS Workload

The 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).

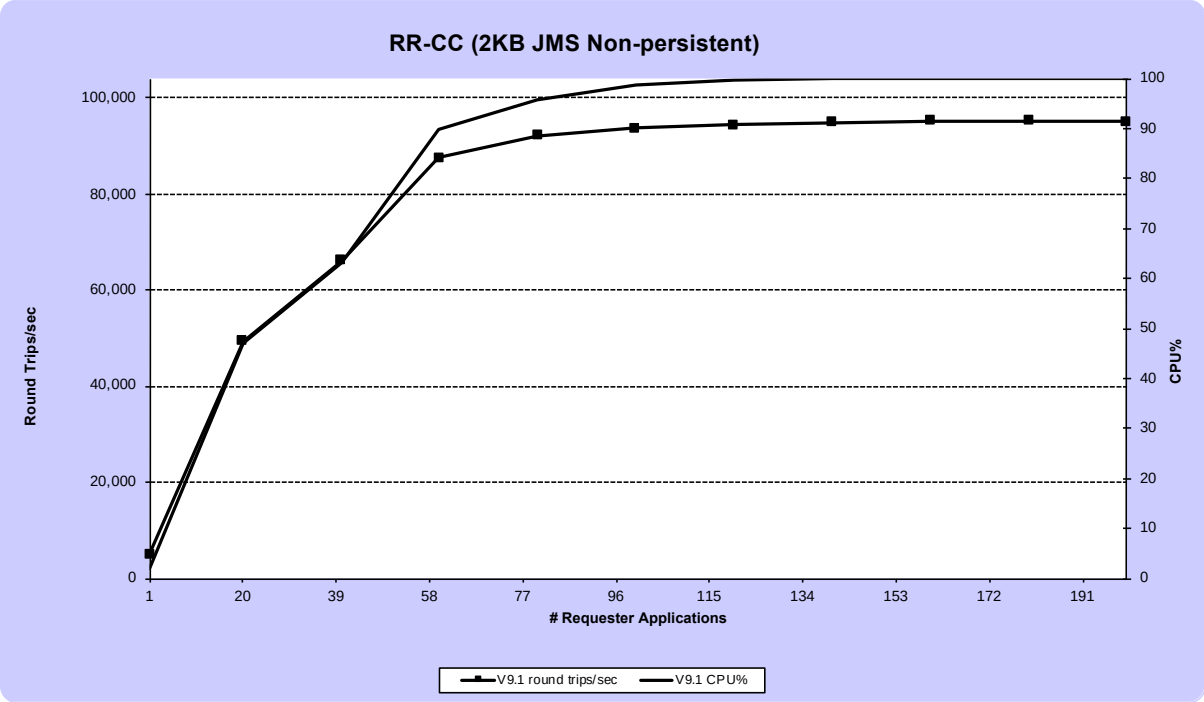

FIGURE 8 - PERFORMANCE RESULTS FOR RR-CC (2KB JMS NON-PERSISTENT)

Once again, the workload exhibits good scaling up to 100% of the CPU (the limiting factor), peaking at approximately 95,000 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 40Gb 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	V9.1 Max Rate*	CPU%	Clients
RR-CC (2KB JMS Non-persistent)	95,166	99.98	180
RR-CC (20KB JMS Non-persistent)	75,376	99.61	150
RR-CC (200KB JMS Non-persistent)	10,499	62.97	100
RR-CC (2MB JMS Non-persistent)	973	57.5	100

***ROUND TRIPS/ SEC**

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

2.3.1 Test setup

Workload type: RR-CC (see section 2).

Message protocol: JMS

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

2.4 RR-CC Workload with TLS (Client mode requesters on separate host. Binding mode responders.)

To illustrate the overhead of enabling TLS, results are provided comparing the performance of the 6 strongest TLS1.2 MQ CipherSpecs, with the baseline client bindings 2KB test presented in section 2.1.

Queue manager authentication is used to setup the TLS conversation.

The TLS 1.2 ciphers under test are shown below (all utilise 256bit encryption, and are FIPS compliant).

CipherSpec	**SuiteB**
TLS_RSA_WITH_AES_256_CBC_SHA256	No
TLS_RSA_WITH_AES_256_GCM_SHA384	No
ECDHE_ECDSA_AES_256_CBC_SHA384	No
ECDHE_ECDSA_AES_256_GCM_SHA384	Yes
ECDHE_RSA_AES_256_CBC_SHA384	No
ECDHE_RSA_AES_256_GCM_SHA384	No

Results for the suite B compliant cipherspec (ECDHE_ECDSA_AES_256_GCM_SHA384) are plotted below. This cipherspec uses a GCM (Galois/Counter Mode) symmetric cipher. Performance testing showed that all GCM based cipherpecs exhibited similar performance. Cipherspecs utilising the older CBC (Chain Block Cipher) symmetric cipher also exhibited similar performance to each other. One CBC cipherspec (ECDHE_ECDSA_AES_256_CBC_SHA384) is included in the plot below, for comparison.

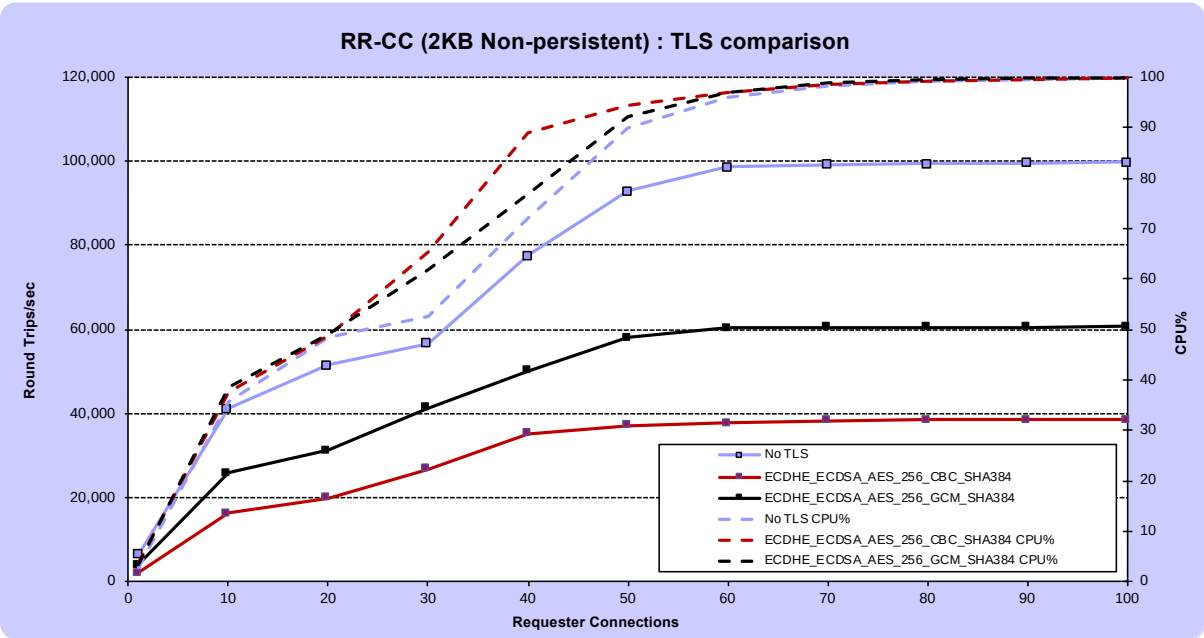

FIGURE 9 - PERFORMANCE RESULTS FOR RR-CC WITH TLS

All tests exhibited good scaling up to 100% of the CPU of the machine. Throughput for GCM based cipherspecs ran at approximately 61% of the throughput of a non-encrypted workload. CBC based cipherspecs exhibited a greater overhead, running at approximately 39% of a non-encrypted workload.

Cipher	V9.1GM Max Rate*	CPU%	Clients
No TLS	99,829	100	100
TLS_RSA_WITH_AES_256_CBC_SHA256	38,441	100	100
TLS_RSA_WITH_AES_256_GCM_SHA384	60,891	100	100
ECDHE_ECDSA_AES_256_CBC_SHA384	38,530	100	100
ECDHE_ECDSA_AES_256_GCM_SHA384	60,736	100	100
ECDHE_RSA_AES_256_CBC_SHA384	38,317	100	100
ECDHE_RSA_AES_256_GCM_SHA384	60,923	100	100

**Round trips/* sec

Table 5 shows the peak rates achieved for all 6 cipherspecs tested, demonstrating the equivalence of performance, based on whether the symmetric key algorithm is CBC, or GCM based.

Cipher	V9.1GM Max Rate*	CPU%	Clients
No TLS	99,829	100	100
TLS_RSA_WITH_AES_256_CBC_SHA256	38,441	100	100
TLS_RSA_WITH_AES_256_GCM_SHA384	60,891	100	100
ECDHE_ECDSA_AES_256_CBC_SHA384	38,530	100	100
ECDHE_ECDSA_AES_256_GCM_SHA384	60,736	100	100
ECDHE_RSA_AES_256_CBC_SHA384	38,317	100	100
ECDHE_RSA_AES_256_GCM_SHA384	60,923	100	100

ROUND TRIPS/ SEC

TABLE 5 - PEAK RATES FOR MQI CLIENT BINDINGS (2KB NON-PERSISTENT) - SSL

2.4.1 Test setup

Workload type: RR-CC (see section 2).

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

3 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 transaction log files. IBM MQ is designed to maximise throughput, regardless of the technology used, by aggregating writes where possible, to the transaction log, where they need to be synchronous to ensure transactional integrity.

The performance of persistent messaging is therefore dependant on the machine hosting MQ, *and* the I/O infrastructure. Some comparisons are shown below between non-persistent and persistent messaging for local storage, and then results for V9.1 in a separate environment (x64 Linux with SAN, SSD & NFS filesystems) are shown to demonstrate the impact of transaction log location.

3.1 RR-BB Workload

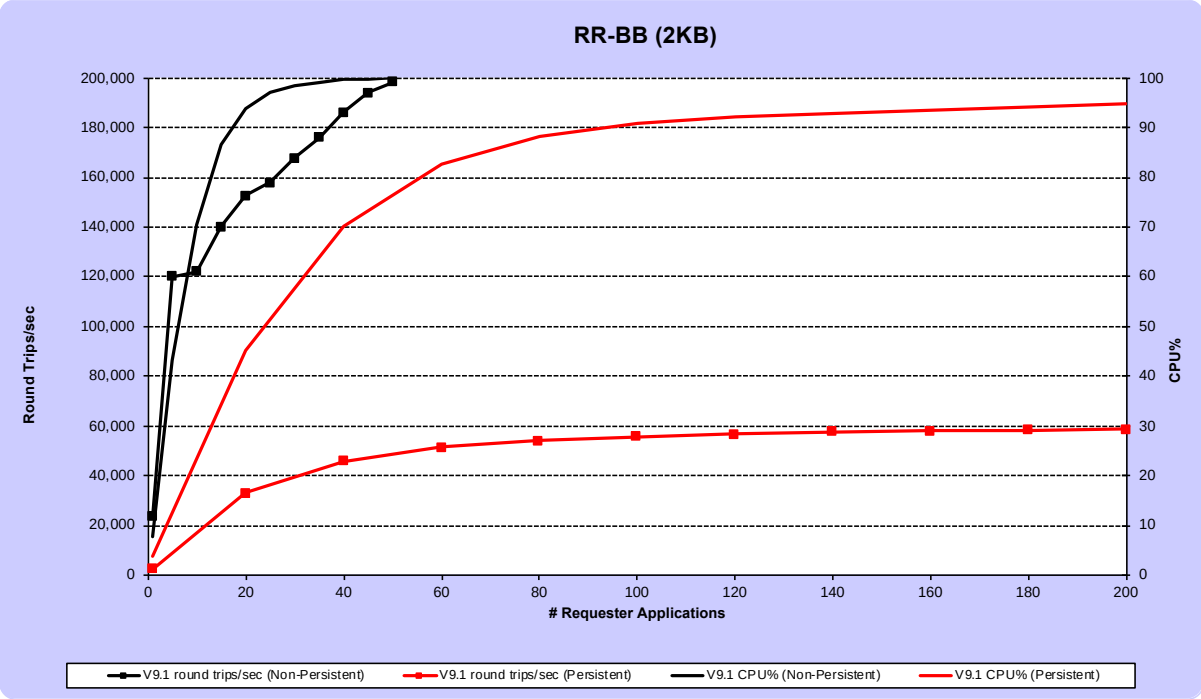

FIGURE 10 - PERFORMANCE RESULTS FOR RR-BB (2KB NON-PERSISTENT VS PERSISTENT)

Figure 10 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.

RR-BB is a variant of RR-CB (see section 1.2) where all applications are connected in bindings mode. This accentuates the impact of persistent messaging since we are no longer limited by network bandwidth.

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 component of IBM MQ to aggregate log data into larger, more efficient writes to disk.

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

Non-persistent workloads are typically limited by the CPU, whilst the transaction log I/O is the limiting factor for the persistent workloads. As the message size goes up, the time spent on the transaction 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.

Test	V9.1 Max Rate*	CPU%	Clients
RR-BB (2K Non-persistent)	198,448	99.96	50
RR-BB (20K Non-persistent)	132,699	99.68	50
RR-BB (200K Non-persistent)	64,741	92.79	24
RR-BB (2MB Non-persistent)	3,249	80.41	20

***ROUND TRIPS/ SEC**

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

Test	V9.1 Max Rate*	CPU%	Clients
RR-BB (2KB Persistent)	58,539	94.75	200
RR-BB (20KB Persistent)	39,050	71.61	100
RR-BB (200KB Persistent)	5,615	19.97	20
RR-BB (2MB Persistent)	634	16.7	10

***ROUND TRIPS/ SEC**

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

3.1.1 Test setup

Workload type: RR-BB (see section 2).

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

3.2 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 transaction logs.

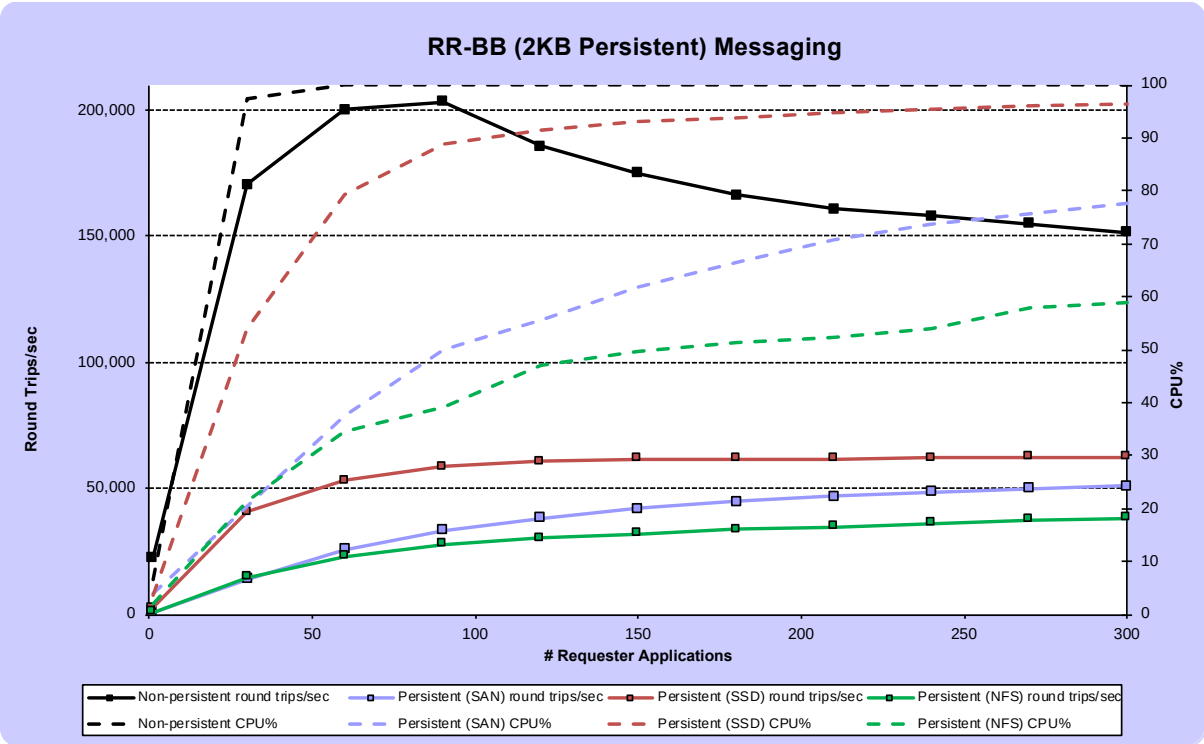

FIGURE 11 - PERFORMANCE RESULTS FOR RR-BB PERSISTENT MESSAGING LOGGING TO SSD, SAN & NFS

To illustrate the impact that the filesystem hosting the transaction logs can have, Figure 11 shows results from running the RR-BB workload with non-persistent and persistent messaging (hosted on different filesystems).

RR-BB is a variant of RR-CB (see section 1.2) where all applications are connected in bindings mode, eliminating the network as a bottleneck (except in the case of NFS, where there is a 10Gb link from the MQ server to the NFS server).

As expected, the non-persistent case is limited by CPU

2.1.1 Test setup

Workload type: RR-BB (see section 2).

Hardware: x64 Linux MQ server, x64 Linux NFS server, IBM SAN Volume Controller, and Storwise V7000 – see section A.2.

Appendix A: Test Configurations

A.1 Hardware/Software – Set1

All of 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:

Lenovo System x3550 M5 – [5463-L2G]

2 x 12 core CPUs.
Core: Intel® Xeon® E5-2690 v3 @ 2.60GHz

128GB RAM

40Gb ethernet adapters connect all three machines via an isolated performance LAN.

A.1.2 Software

Red Hat Enterprise Linux Server release 7.4 (Maipo)

JMSPerfHarness test driver (see Appendix C:)

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

IBM MQ V9.1

A.2 Hardware/Software – Set2 (Persistent messaging comparisons)

The persistent messaging tests in section 3.2, comparing different filesystems used to host the MQ transaction logs, were run with the following hardware/software.

A.2.1 Hardware

The MQ Server and NFS Server are two identical machines:

Lenovo System x3550 M5 – [8869-AC1]

2 x 14 core CPUs.
Core: Intel® Xeon® E5-2690 v4 @ 2.60GHz

128GB RAM

10Gb ethernet adapters connected the QM server to the NFS server, via an isolated performance LAN.

The SAN test used an IBM Storwize V7000 populated with 10,000 rpm disks configured in a RAID 10 array, and fronted by an IBM SAN Volume Controller (SVC) with 20GB of RAM. The SVC was connected to the MQ server via a dual-port 8Gb fibre channel adapter.

A.2.2 Software

Red Hat Enterprise Linux Server release 7.4 (Maipo)

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

IBM MQ V9.1

A.3 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.3.1 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 = 13121479
net.ipv4.ip_local_port_range = 1024 65535
vm.max_map_count = 1966080
kernel.pid_max = 655360
kernel.sem = 1000 1024000 500 8192
kernel.msgmnb = 131072
kernel.msgmax = 131072
kernel.msgmni = 32768
kernel.shmmni = 8192
kernel.shmall = 4294967296
kernel.shmmax = 137438953472
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
@mqm soft core unlimited
@mqm hard core unlimited
```

A.3.2 IBM MQ

The following parameters are added or modified in the qm.ini files for the tests run in section 2 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.

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/

Linear Logging Improvements Blog Article
https://www.ibm.com/developerworks/community/blogs/messaging/entry/Logger_enhancements_for_MQ_v9_0_2?lang=en

Implicit Syncpoint Blog Article
https://developer.ibm.com/messaging/2018/04/24/implicit-syncpointing-persistent-messages-put-outside-syncpoint

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

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

Persistent Messaging Performance Paper
https://ibm-messaging.github.io/mqperf/mqio_v1.pdf

Transaction Log Sizing Documentation
https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.con.doc/q018470_.htm