WebSphere MQ for Windows V5.3 Performance Evaluations for Windows XP Professional Version 1.0

4 November 2002

Kevin Aires

WebSphere MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

Property of IBM

Notices

This report is intended to help the customer perform capacity planning. The information is not intended as the specification of any programming interfaces that are provided by WebSphere MQ.

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which it operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed “as-is”. The use of this information and the implementation of any of the techniques is the responsibility of the customer. Much depends on the ability of the customer to evaluate the data and project the results to their operational environment.

The performance data contained in this report was measured in a controlled environment and the results obtained in other environments may vary significantly.

Trademarks and service marks

The following terms used in this publication are trademarks of the IBM Corporation in the United States or other countries or both:

IBM

MQSeries

WebSphere MQ

SupportPac

FFST

AIX

Microsoft, Windows, Windows NT, Windows 2000, Windows XP Professional and .net are trademarks of Microsoft Corporation in the United States, other countries, or both.

First edition - November 2002.
This edition for Windows XP Professional applies to V1.0 of WebSphere MQ for Windows V5.3 – Performance Evaluations and to all subsequent releases and modifications until otherwise indicated in new editions.
(C) Copyright International Business Machines Corporation 2002. 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.

Preface

This report presents the results of performance evaluations of WebSphere MQ for Windows V5.3, and is intended to assist with capacity planning. An IBM Netfinity 8500R machine (4-way 700 MHz CPU, 8 GB RAM) running Windows XP Professional was used as the device under test for all the measurements in this report. **Note, however, that current versions of Windows XP Professional do not recognise more than 2CPUs or more than 4GB of RAM, hence only 2 CPUs and 4GB of RAM were utilised. This means that some performance results may appear lower than for Windows 2000 Advanced Server in other reports.** This report gives an initial look at performance of Windows XP Professional before future versions of Windows XP Professional, which will support more processors, are released under the ".net" brand. For full details of the measurement environment see page 32.

Target Audience

This SupportPac is designed for people who:

- Will be designing and implementing environments using WebSphere MQ for Windows V5.3.
- Want to understand the performance limits of WebSphere MQ for Windows V5.3.
- Want to understand how to tune WebSphere MQ for Windows V5.3.
- Want to understand the implications of using Windows XP to run Websphere MQ 5.3.

Readers should have a general awareness of the Windows XP operating system and of WebSphere MQ (formerly MQSeries) in order to make best use of this SupportPac. Readers should read the section **How this document is arranged** to familiarise themselves with the layout of this report.

Contents of this SupportPac

- Charts which summarise the performance highlights of this release.
- Charts and tables which summarise the performance characteristics of various WebSphere MQ client, distributed, and local queue manager configurations.
- Interpretation of the measurements and their implications for designing or sizing WebSphere MQ client, distributed, and local queue manager configurations.

Feedback on this SupportPac

We welcome constructive 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 manner more useful to you? Please direct any comments of this nature to: **WMQPG@uk.ibm.com**.

Specific queries about performance problems on your WebSphere MQ system should be directed in the first instance to your local IBM MQ sales representative.

Acknowledgements

The author is very grateful to Lucas Partridge and Richard Eures for their help in producing this report.

How this document is arranged

Release highlights

Page: 1

Outlines the performance achieved using WebSphere MQ V5.3 on Windows XP. The highlights are a subset of the results shown in the performance headlines section.

Performance headlines

Page: 4

Contains the performance headlines for each of the following three scenarios, with MQI driving applications connected:

- to a local queue manager, or
- to a remote queue manager over MQI-client channels, or
- to a local queue manager, driving throughput between the local and remote queue manager over server channel pairs.

The headline tests show:

- the maximum message throughput achieved with an increasing number of MQI applications,
- the maximum number of MQI-clients that could be connected to a queue manager,
- the maximum number of applications that could be connected before the average response time exceeded one second, using a fixed number of channel pairs.

Large messages

Page: 17

Contains performance measurements for large messages. This includes MQI response times for 50 byte to 2 MB messages, and also for 20 KB and 200 KB messages using the same scenarios as for the performance headlines.

Trusted server application

Page: 25

Contains performance measurements for a trusted server application, using the same three scenarios as for the performance headlines.

Short sessions

Page: 26

Contains performance measurements for short sessions. A short session is a session in which an MQI application processes only a few messages between connecting to and disconnecting from the queue manager.

CONTENTS

TABLES

FIGURES

1 Release highlights

Unless otherwise stated, all the measurements described in this report were conducted using messages with 2 KB (2048 bytes) of application data and the application configuration described in **8.3.2 Scenario workload**. For diagrams of the workload scenarios used see **Figure 4**, **Figure 7** and **Figure 12**.

1.1 Peak throughput highlights

- Peak nonpersistent message throughput was:
  - 4557 messages/s in a local queue manager environment.
  - 3084 messages/s in a MQI client environment.
  - 3617 messages/s in a distributed queuing environment.
- Peak persistent message throughput was:
  - 1014 messages/s in a local queue manager environment
  - 886 messages/s in a MQI client environment.
  - 767 messages/s in a distributed queuing environment.

1.2 Local queue manager – peak message throughput

Figure 1 below shows the peak throughput achieved for nonpersistent and persistent 2 KB messages with a local queue manager.

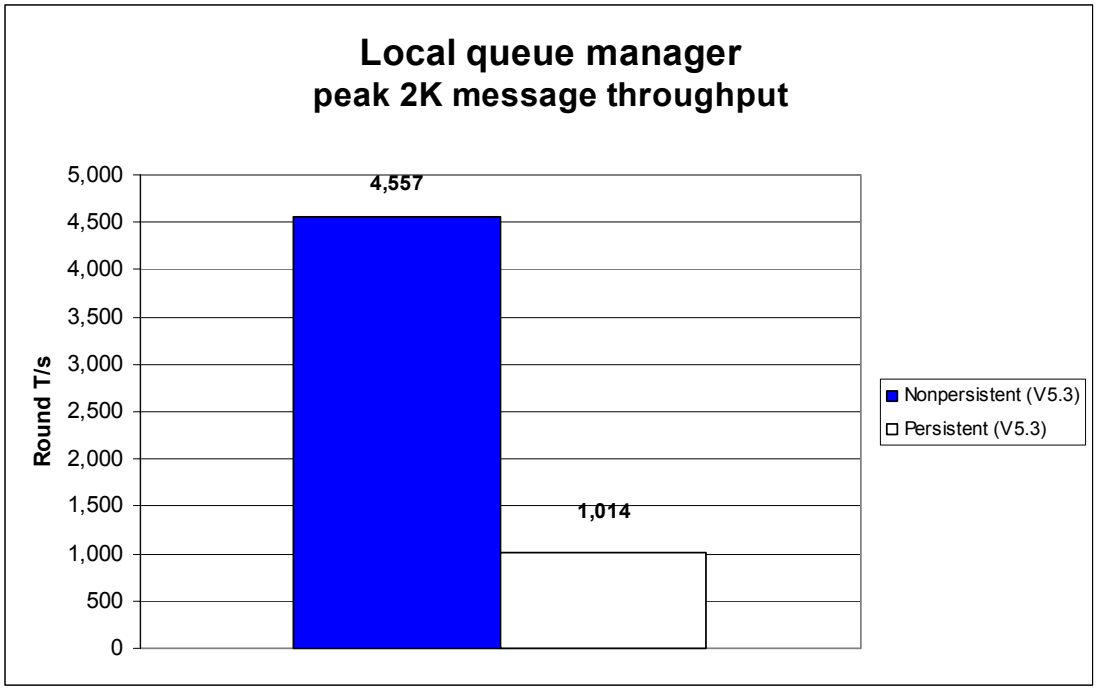

Figure 1 – Peak 2 KB message throughput, local queue manager

1.3 Client channels – peak message throughput

Figure 2 below shows the peak message throughput achieved for nonpersistent and persistent messages with MQI-client channels.

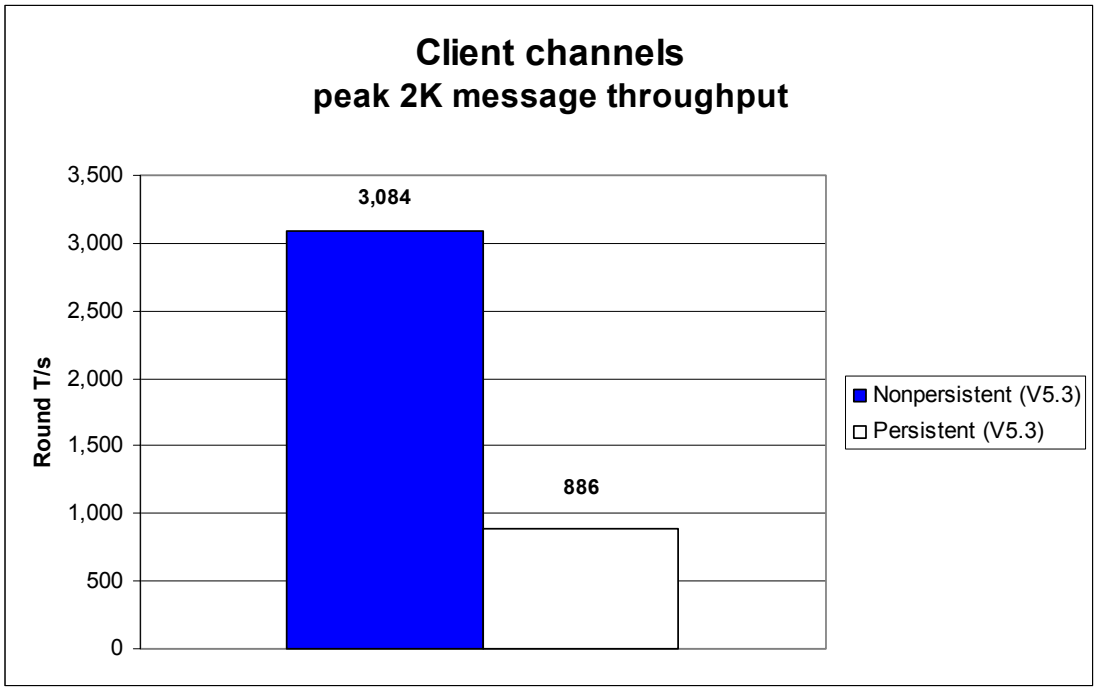

Figure 2 – Peak 2 KB message throughput, client channels

1.4 Distributed queuing – peak message throughput

Figure 3 below shows the peak message throughput achieved for nonpersistent and persistent messages with server channels. Note that throughput was still increasing with the number of driving applications for the persistent measurement so caution is advised in interpreting this peak figure shown (see also **Figure 14**).

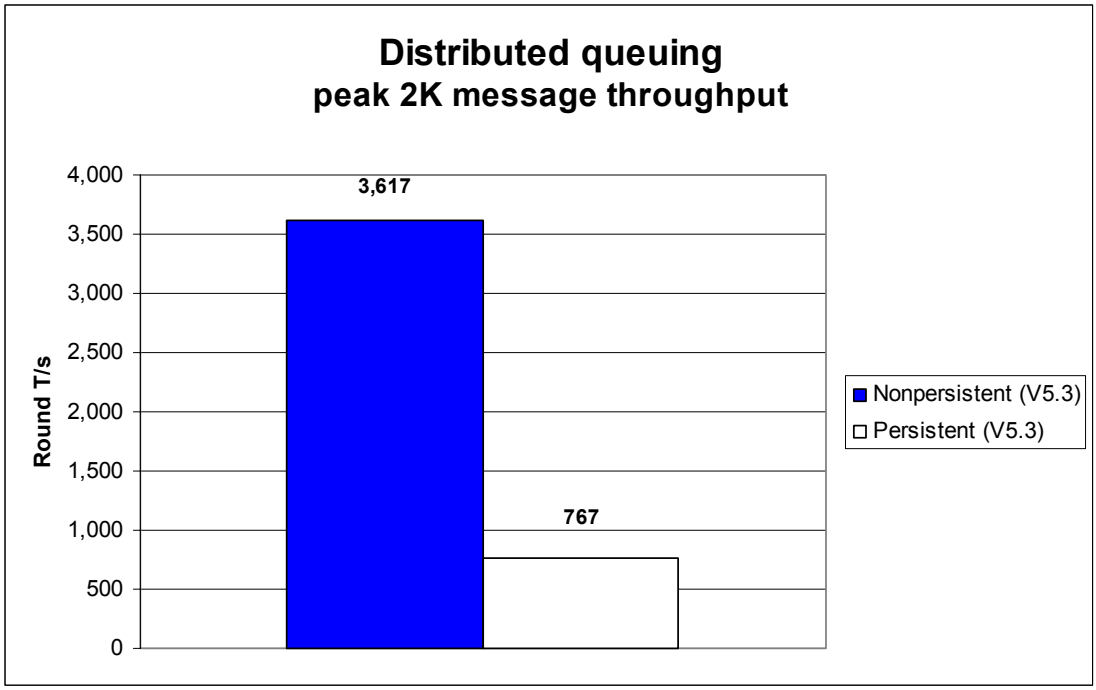

Figure 3 – Peak 2 KB message throughput, distributed queuing

2 Performance headlines

2.1 Local queue manager scenario

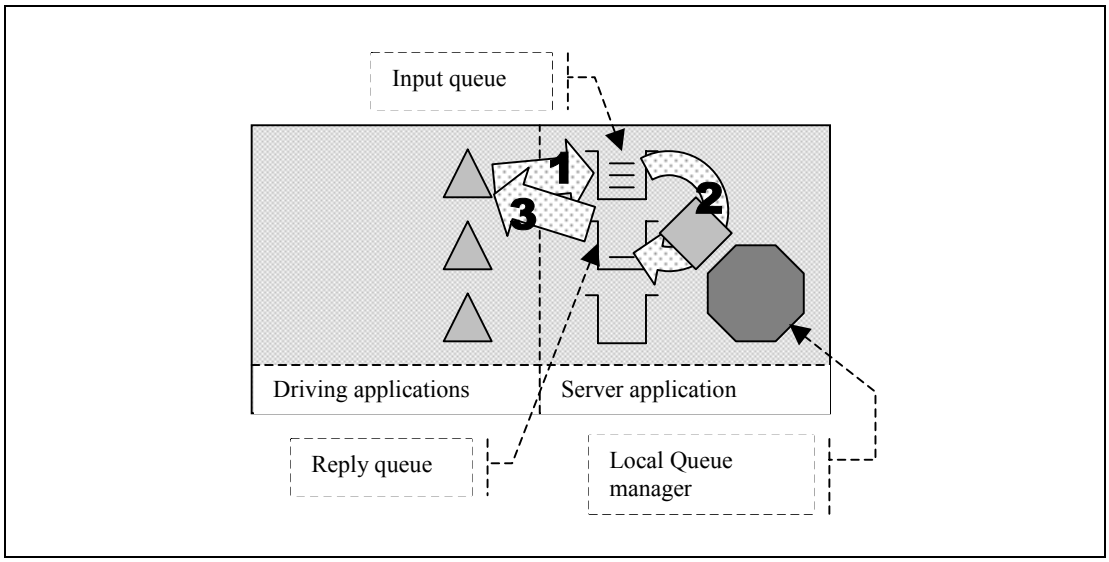

Figure 4 – Connections into a local queue manager

1) The driving application puts a request message onto the common input queue attached to the local queue manager, and records the message's message ID assigned by the queue manager. The driving application then waits indefinitely for a reply to arrive on the common reply queue.

2) The server application gets messages from the common input queue and places a reply on the common reply queue. The queue manager copies the message ID from the request message into the correlation ID field of the reply message.

3) The driving application gets a reply from the common reply queue using the message ID recorded from the corresponding request message as the correlation ID in the message descriptor. The driving application then either puts another request message immediately, or it waits until a specified think time has elapsed since it put the last request message.

Figure 5 and **Figure 6** show the peak nonpersistent and persistent message throughputs achieved using the local queue manager scenario in **Figure 4** above. Zero think-time was used in order to achieve maximum throughput with as few driving applications as possible. The throughput for the local tests was compared to running the same tests on Windows 2000 Advanced Server running with only 2 processors (note Windows XP Professional only supports up to 2 processors).

2.1.1 Nonpersistent messages – local queue manager

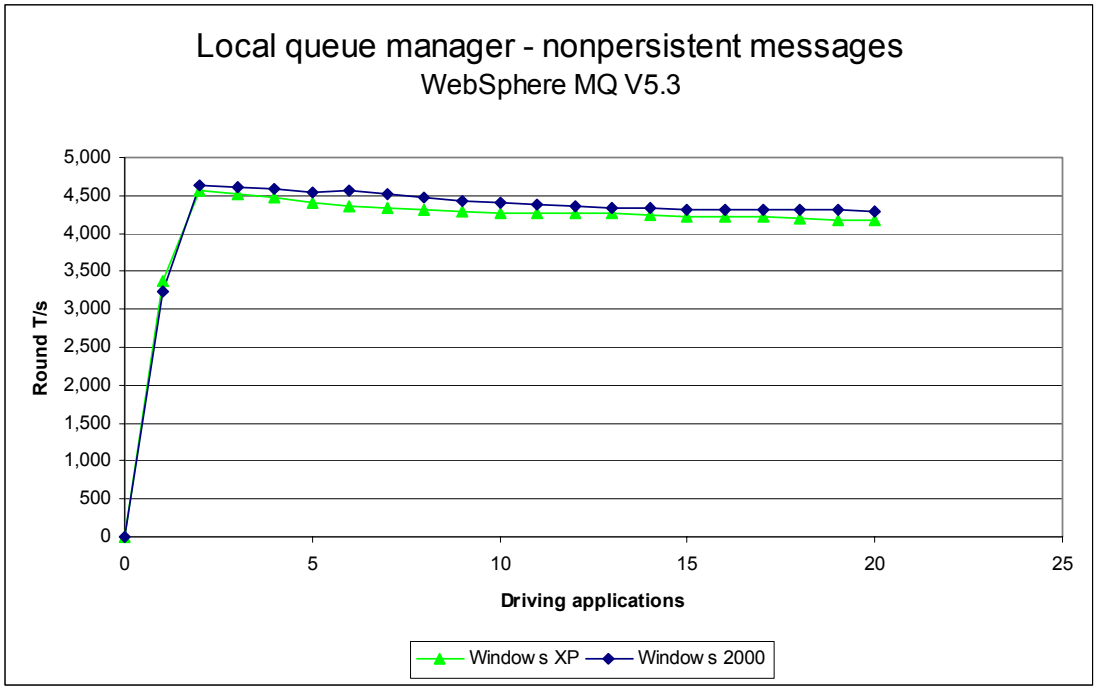

Figure 5 – Performance headline, nonpersistent messages, local queue manager

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3 (WindowsXP)	2	4557	0.001

Table 1 – Performance headline, 2 KB nonpersistent messages, local queue manager (test name 'local_np1')

Figure 5 and **Table 1** shows that the peak throughput occurred at 2 applications for Windows XP Professional and then slightly decreased and levelled out to around 4250 round trips/s. Refer to the **Glossary** for a description of each of the table column headings.

Figure 5 also demonstrates the very similar performance of Websphere MQ V5.3 running on Windows 2000 (running with 2 processors) and Windows XP Professional (which only recognises 2 processors).

2.1.2 Persistent messages – local queue manager

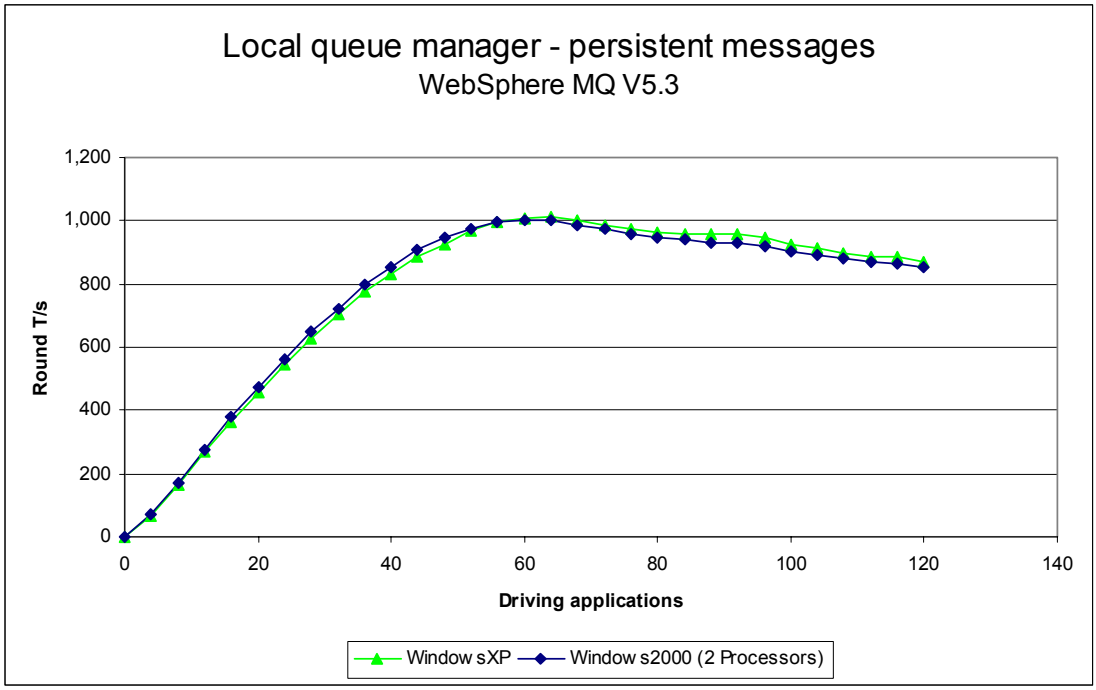

Figure 6 - Performance headline, persistent messages, local queue manager

Product version	Apps	Round trips/s	Response Time (s)
WebSphere MQ V5.3	64	1014	0.071

Table 2 – Performance headline, 2 KB persistent messages, local queue manager (test name 'local_pm3')

Figure 6 shows that the peak throughput occurred at 64 applications and then gradually declined.

Figure 6 also demonstrates the very similar performance of Websphere MQ V5.3 running on Windows 2000 (running with 2 processors) and Windows XP Professional (which only recognises 2 processors).

2.2 Client channels scenario

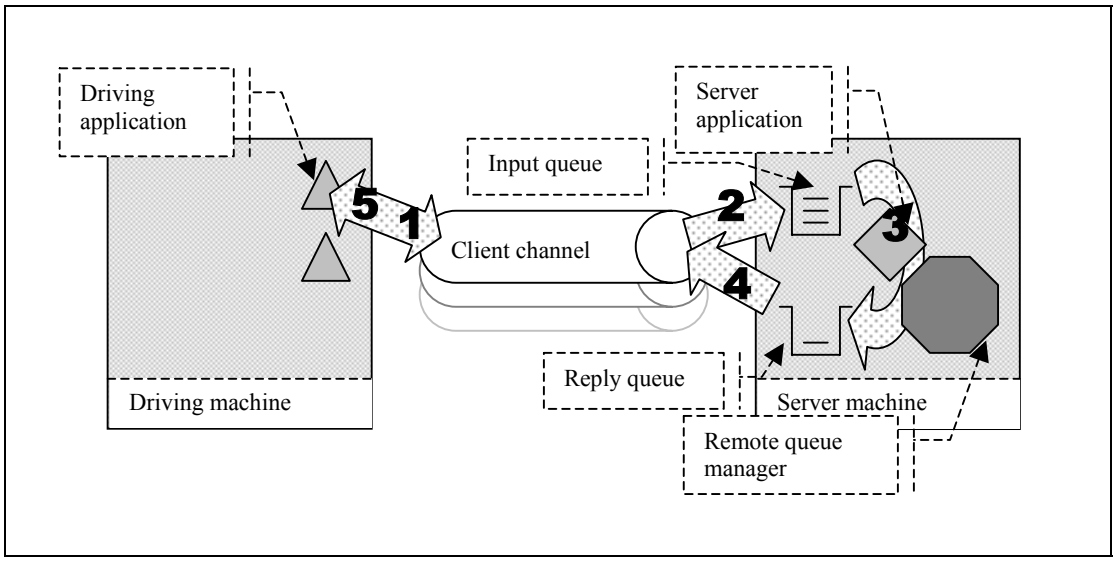

Figure 7 – MQI-client channels into a remote queue manager

1, 2) The driving application puts a request message (over a client channel) to the common input queue attached to a remote queue manager; and records the message's message ID assigned by the queue manager. The driving application then waits indefinitely for a reply to arrive on the common reply queue.

3) The server application gets messages from the common input queue and places a reply on the common reply queue. The queue manager copies the message ID from the request message into the correlation ID field of the reply message.

4,5) The driving application gets a reply (over the client channel) from the common reply queue using the message ID recorded from the corresponding request message as the correlation ID in the message descriptor. The driving application then either puts another request message immediately, or it waits until a specified think time has elapsed since it put the last request message.

Figure 8 and **Figure 9** below show the peak nonpersistent and persistent message throughputs achieved using the client channels scenario in **Figure 7** above. Zero think-time was used in order to achieve maximum throughput with as few driving applications as possible.

2.2.1 Nonpersistent messages – client channels

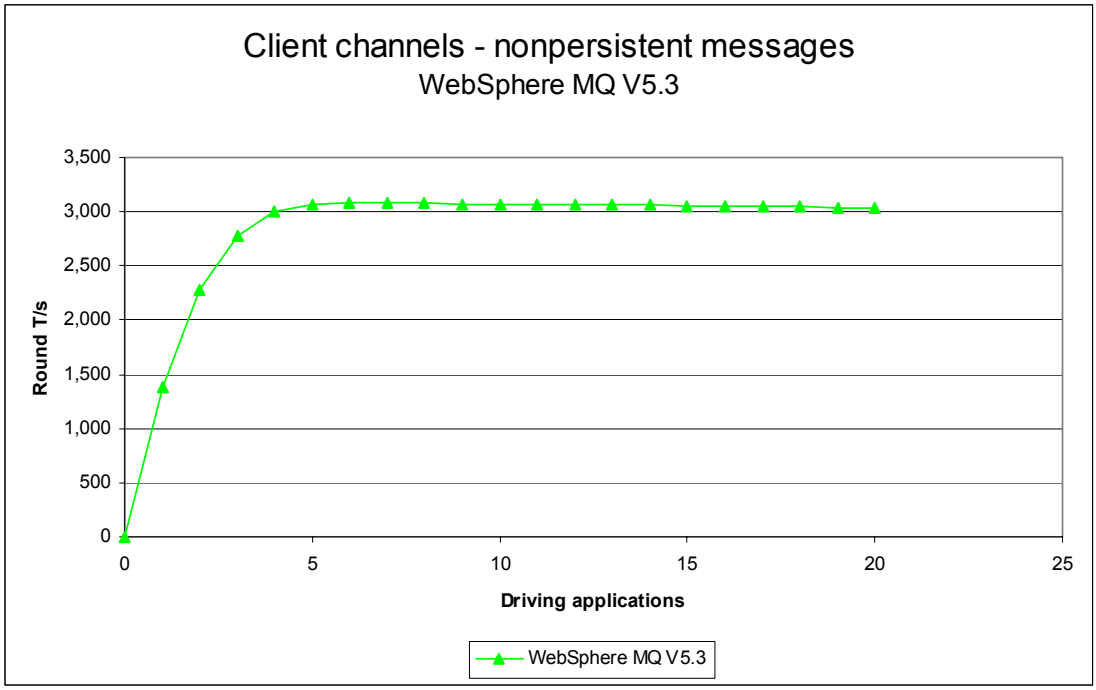

Figure 8 – Performance headline, nonpersistent messages, client channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	6	3084	0.002

Table 3 – Performance headline, 2 KB nonpersistent messages, client channels (test name 'clnp1')

Figure 8 shows that the number of round trips/s increased with the number of driving applications until it reached 6 applications, when it levelled off at around 3000 round trips/s.

2.2.2 Persistent messages – client channels

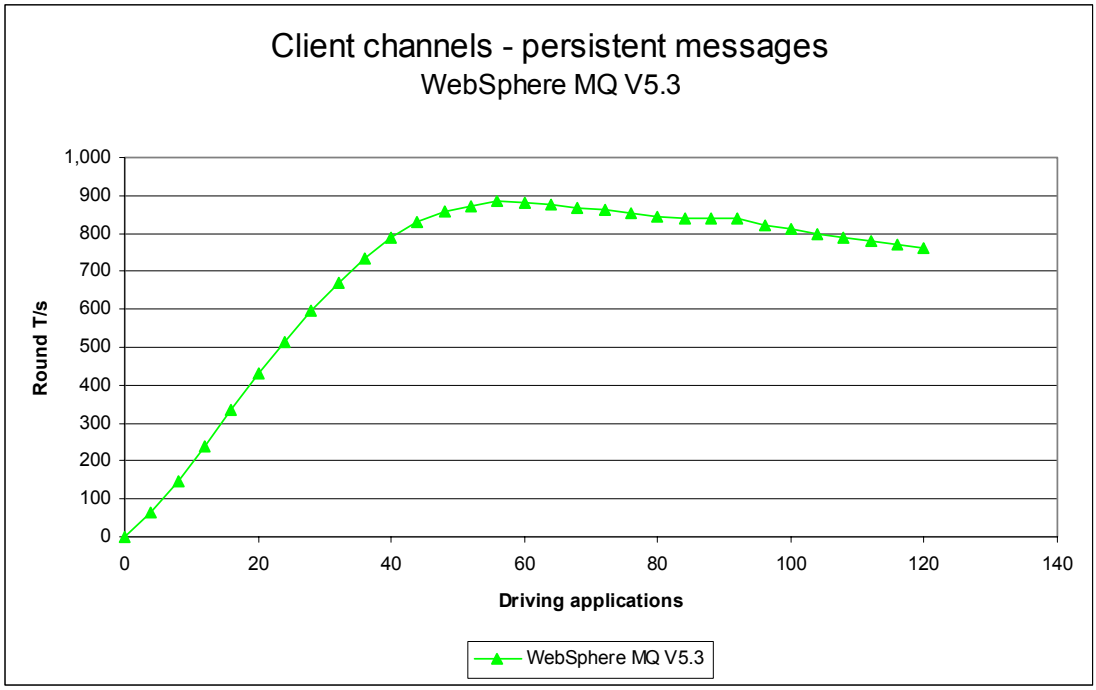

Figure 9 – Performance headline, persistent messages, client channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	56	886	0.070

Table 4 – Performance headline, 2 KB persistent messages, client channels (test name 'clpm3')

Figure 9 shows that the number of round trips increased steadily with the number of driving applications until it reached a peak at 56 applications, after which the number of round trips/s gradually declined.

2.2.3 Rated client channel tests

For the following client channel measurements the message rate used was one round trip per driving application per second. Thus a driving application would not send another request message until it had received the reply to its previous request *and* at least one second had elapsed since it had sent the previous request. The purpose of such tests was to see how many driving applications could be connected before average response time exceeded one second.

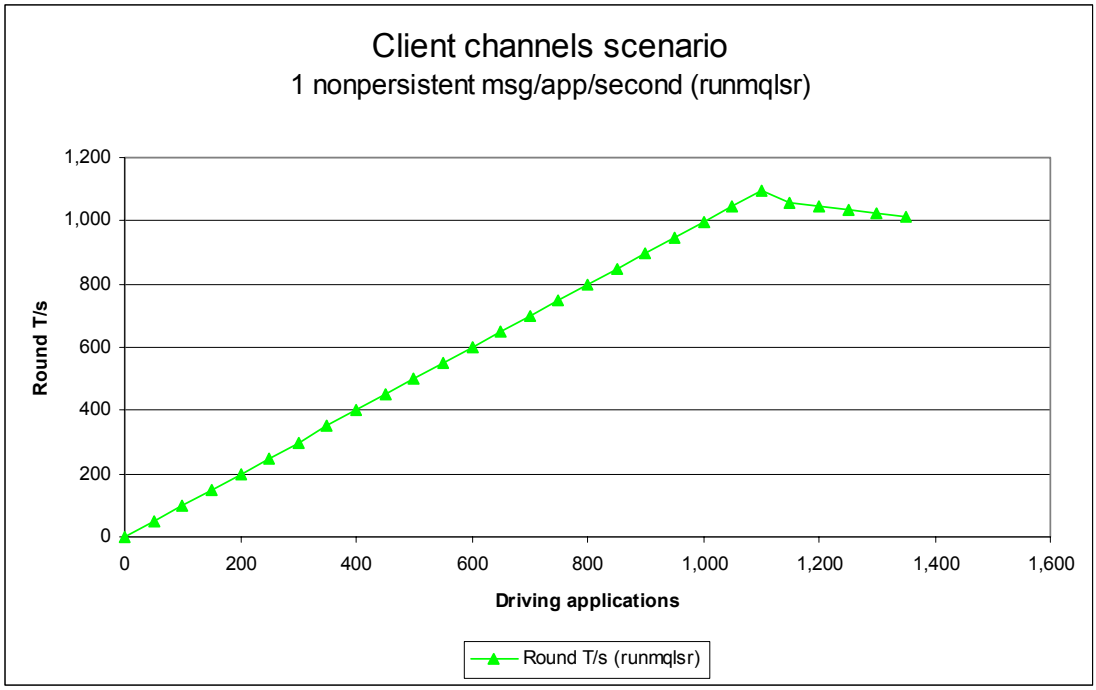

Figure 10 – Rated test, nonpersistent messages, client channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	1,100	1,098	0.020

Table 5 – One message per driving application per second, 2 KB nonpersistent messages, client channels (test name 'clnp1_r3600_runmqlsr')

Figure 10 shows that the test became constrained at 1150 applications where the average response time exceeded 1 second.

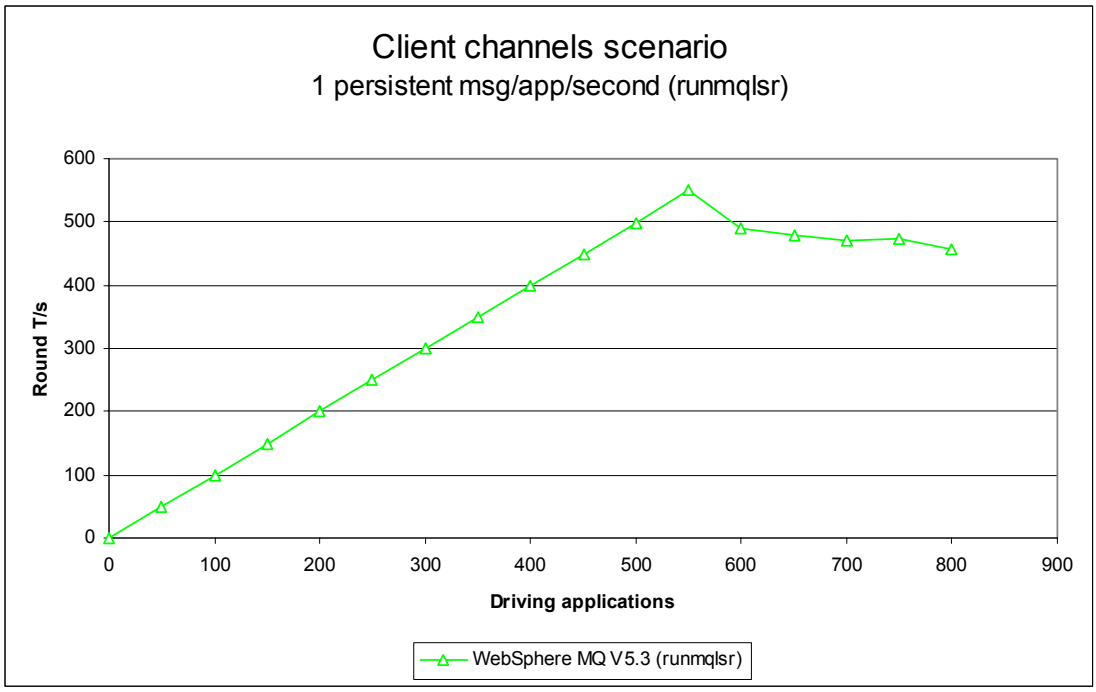

Figure 11 – Rated test, persistent messages, client channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	550	550	0.079

Table 6 – One message per driving application per second, 2 KB persistent messages, client channels (test name 'clpm3_r3600_runmqlsr')

Figure 11 shows that the test became constrained at 600 applications, where the response time exceeded 1 second.

2.3 Distributed queuing scenario

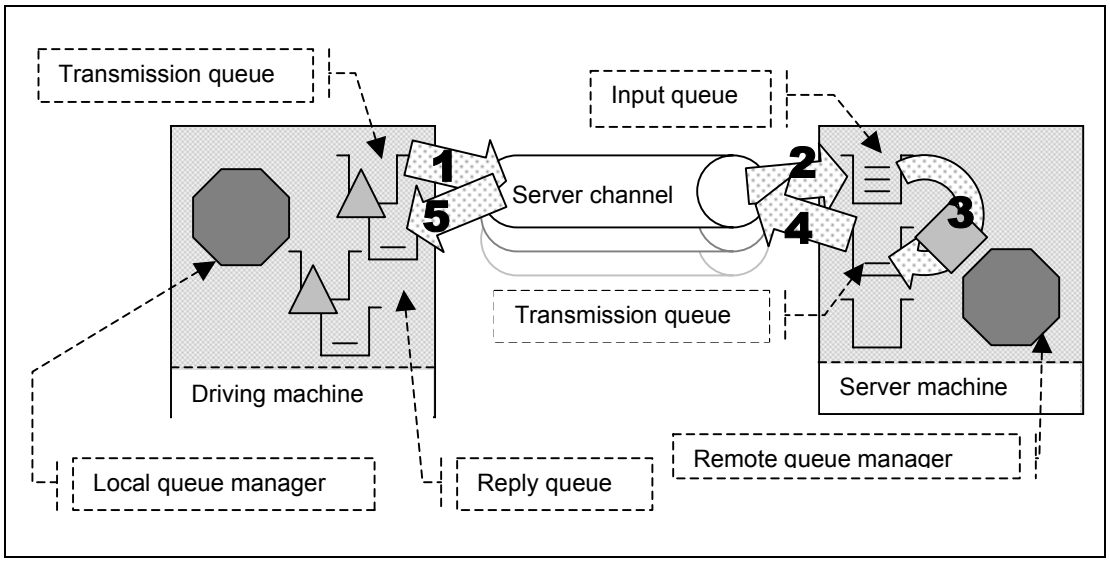

Figure 12 – Server channels between two queue managers

1) The driving application puts a message to a local definition of the remote common input queue attached to a remote queue manager, and records the message's message ID assigned by the local queue manager. The driving application then waits indefinitely for a reply to arrive on the common reply queue.

2) The message channel agent takes messages off the channel and places them on the common input queue on the server machine.

3, 4) The server application gets messages from the common input queue and places a reply on the common reply queue. The queue manager copies the message ID from the request message into the correlation ID field of the reply message.

5) The driving application gets a reply from the common reply queue using the message ID recorded from the corresponding request message as the correlation ID in the message descriptor. The driving application then either puts another request message immediately, or it waits until a specified think time has elapsed since it put the last request message.

Figure 13 and **Figure 14** below show the peak nonpersistent and persistent message throughputs achieved using the distributed queuing scenario in **Figure 12** above. Zero think-time was used in order to achieve maximum throughput with as few driving applications as possible.

2.3.1 Nonpersistent messages – server channels

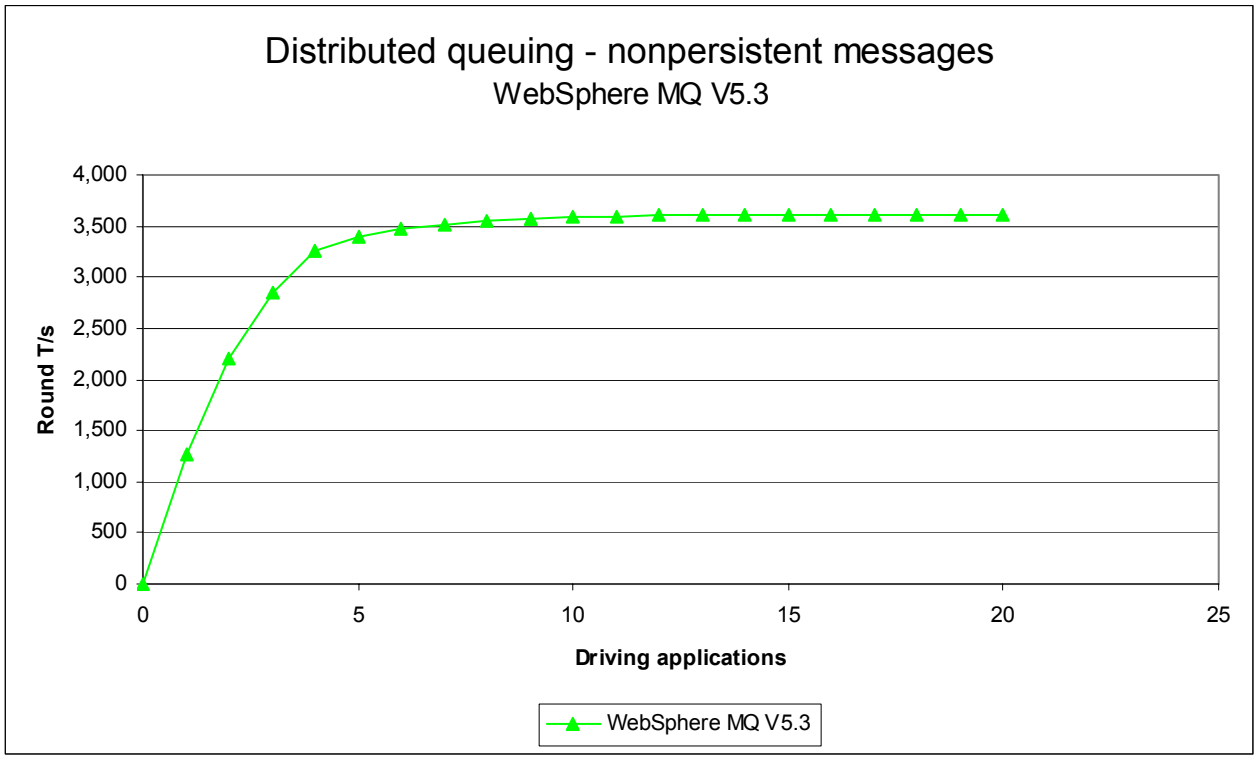

Figure 13 – Performance headline, nonpersistent messages, server channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	18	3617	0.007

Table 7 – Performance headline, 2 KB nonpersistent messages, server channels (test name 'dqnp1')

4 channel pairs were used for the nonpersistent test. **Figure 13** shows that the maximum round trips occurred when using 18 applications, but that throughput was very similar from about 8 applications onwards.

2.3.2 Persistent messages – server channels

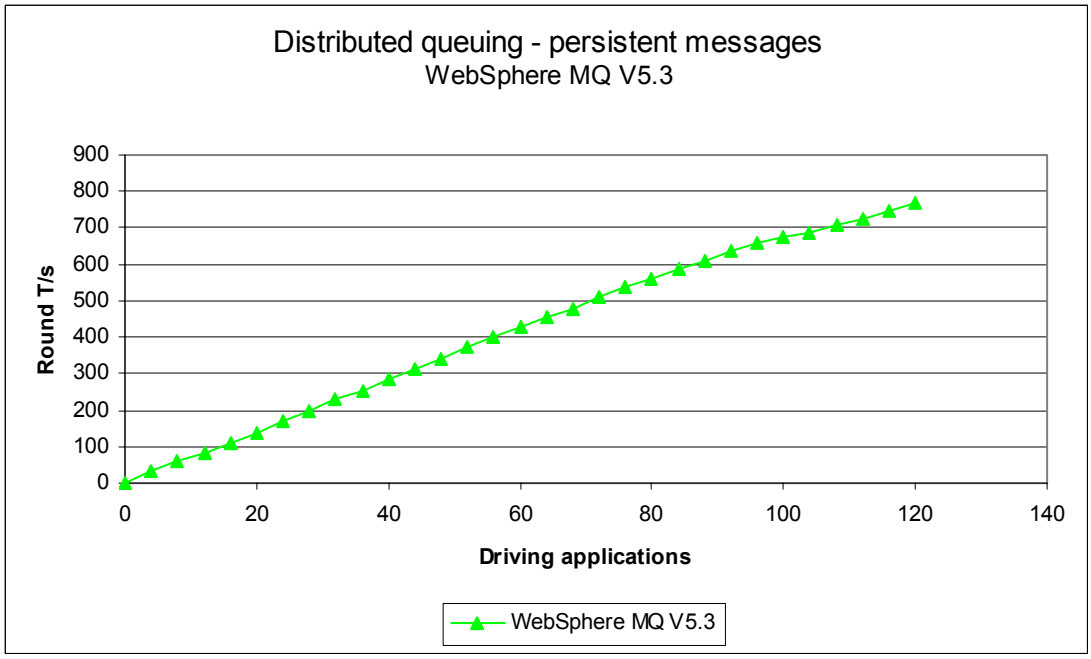

Figure 14 – Performance headline, persistent messages, server channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	120	767	0.180

Table 8 – Performance headline, 2 KB persistent messages, server channels (test name 'dqpm1')

Note that throughput was still increasing with the number of driving applications when the tests were completed at 120 applications. 2 channel pairs were used for the persistent test.

2.3.3 Rated server channel tests

For the following distributed queuing measurements, the rate used was one round trip per driving application per second. The purpose of such tests was to see how many driving applications could be connected before average response time exceeded one second. A fixed number of four and two server channel pairs were used for the nonpersistent and persistent message tests respectively.

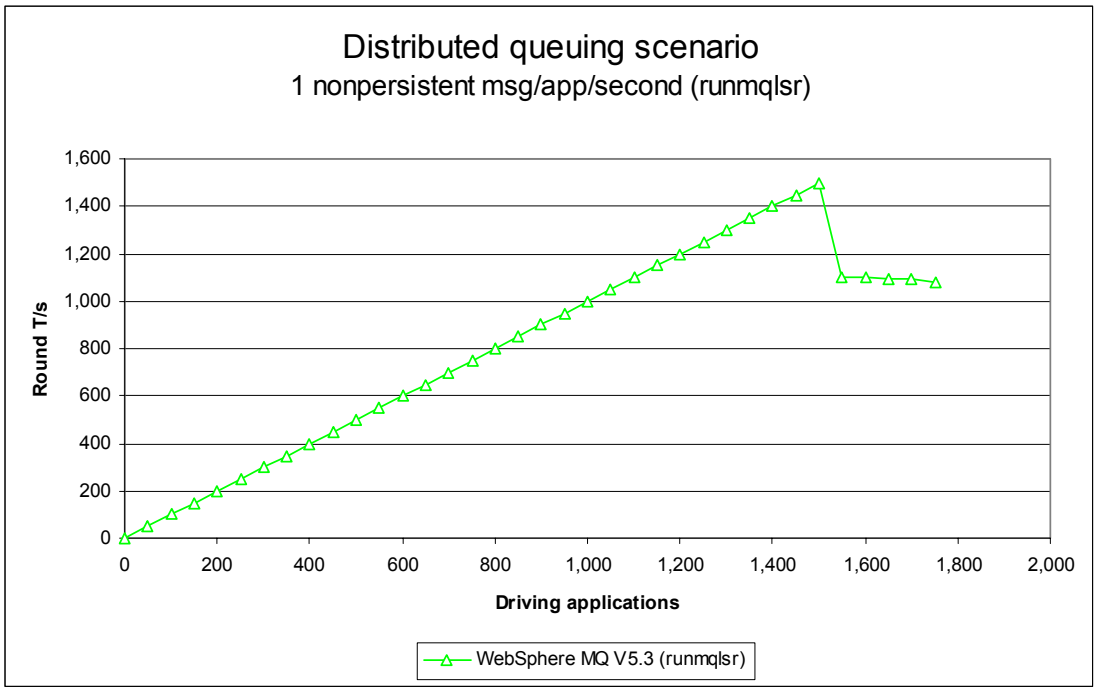

Figure 15 – Rated test, nonpersistent messages, server channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	1,500	1,500	0.026

Table 9 – One message per driving application per second, 2 KB nonpersistent messages, server channels (test name 'dqnp1_r3600_runmqlsr')

Figure 15 shows that the test became constrained at 1550 applications, where the response time exceeded 1 second.

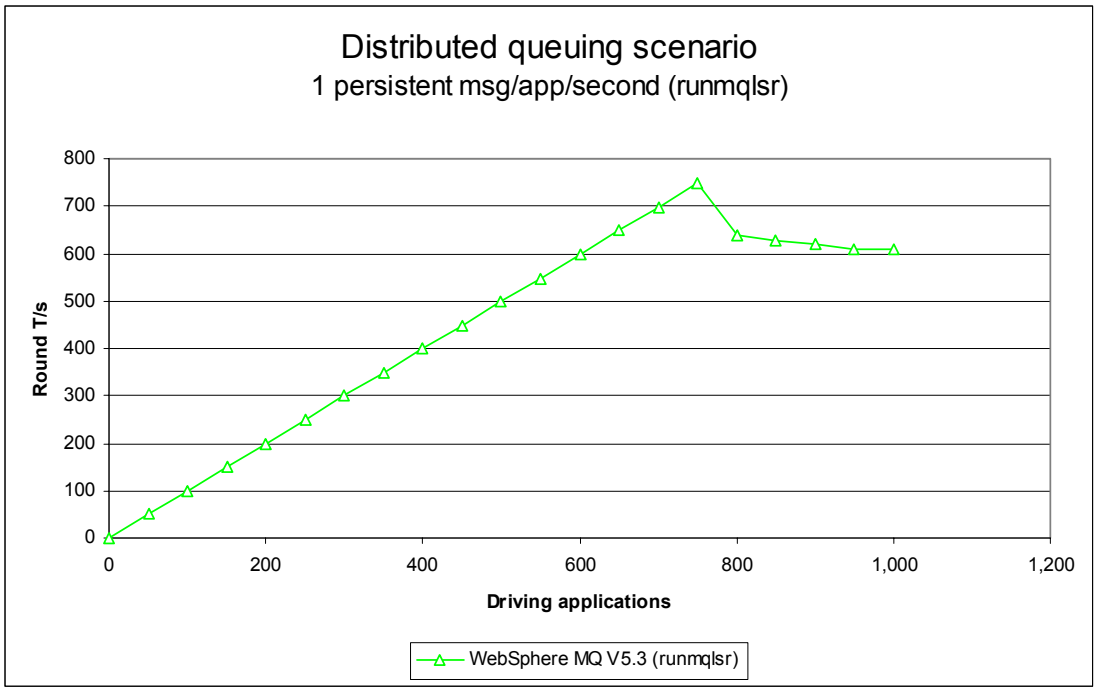

Figure 16 – Rated test, persistent messages, server channels

Product version	Apps	Round trips/s	Response time (s)
WebSphere MQ V5.3	750	750	0.199

Table 10 – One message per driving application per second, 2 KB persistent messages, server channels (test name 'dqpm1_r3600_runmqlsr')

Figure 16 shows that the test became constrained at 800 applications, where the response time exceeded 1 second.

3 Large messages

3.1 MQI response times (50 bytes to 2 MB) – local queue manager

These measurements were conducted using a single-threaded application putting and getting messages to an empty queue attached to a local queue manager (see **8.3.1 MQI performance tool** for details). Each point on these four charts represents the 90th percentile of 5000 separate measurements. To facilitate reading of the charts the results for both nonpersistent and persistent messages are shown first for 50 bytes to 32 KB and then for 32 KB to 2 MB.

See **Test Descriptions** on **page 34** for test configuration of large message tests.

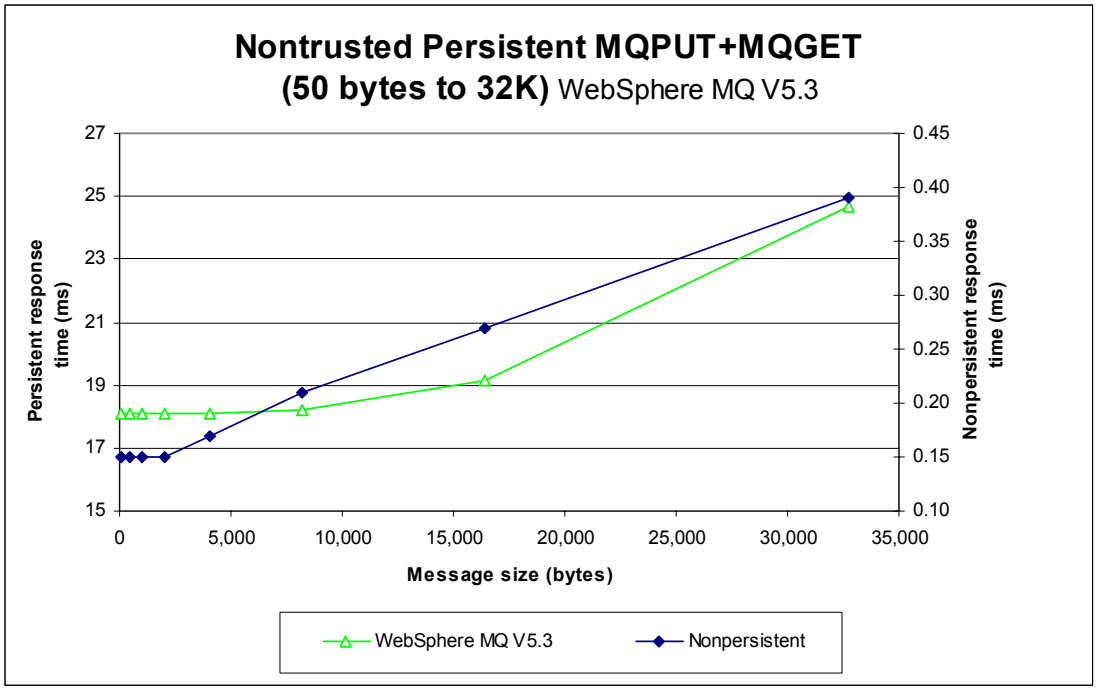

Figure 17 – The effect of message size on MQI response time (50 bytes to 32 KB)

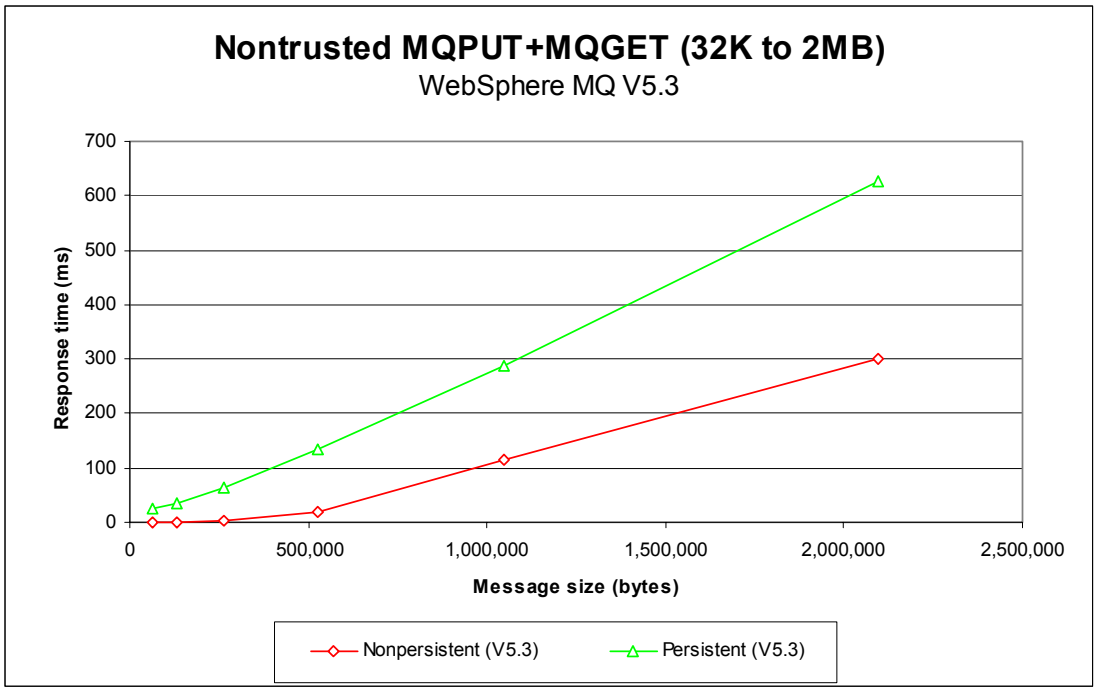

Figure 18 – The effect of message size on MQI response time (32KB to 2 MB)

3.2 Large messages (20 and 200 KB) – local queue manager

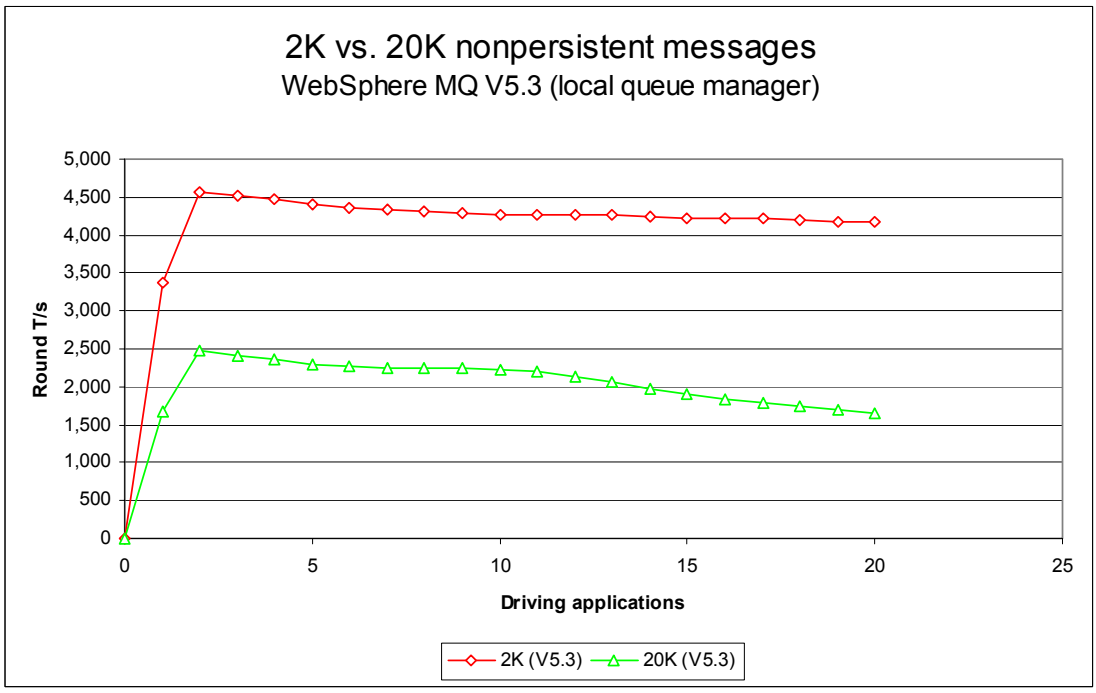

Figure 19 – 2 and 20 KB nonpersistent messages, local queue manager

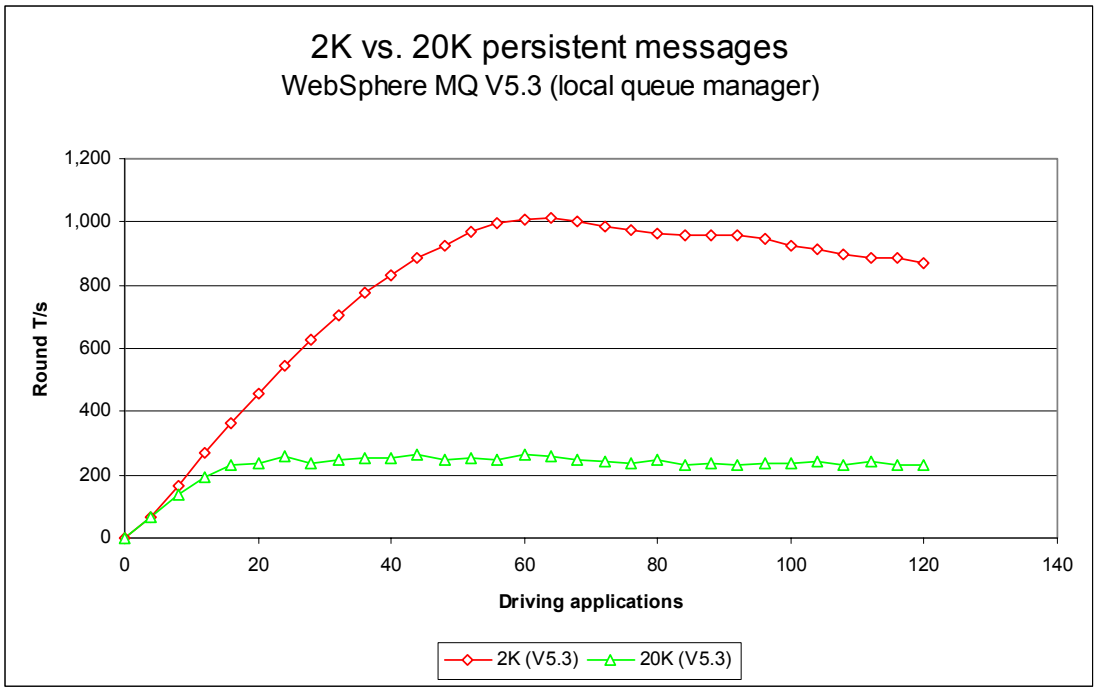

Figure 20 – 2 and 20 KB persistent messages, local queue manager

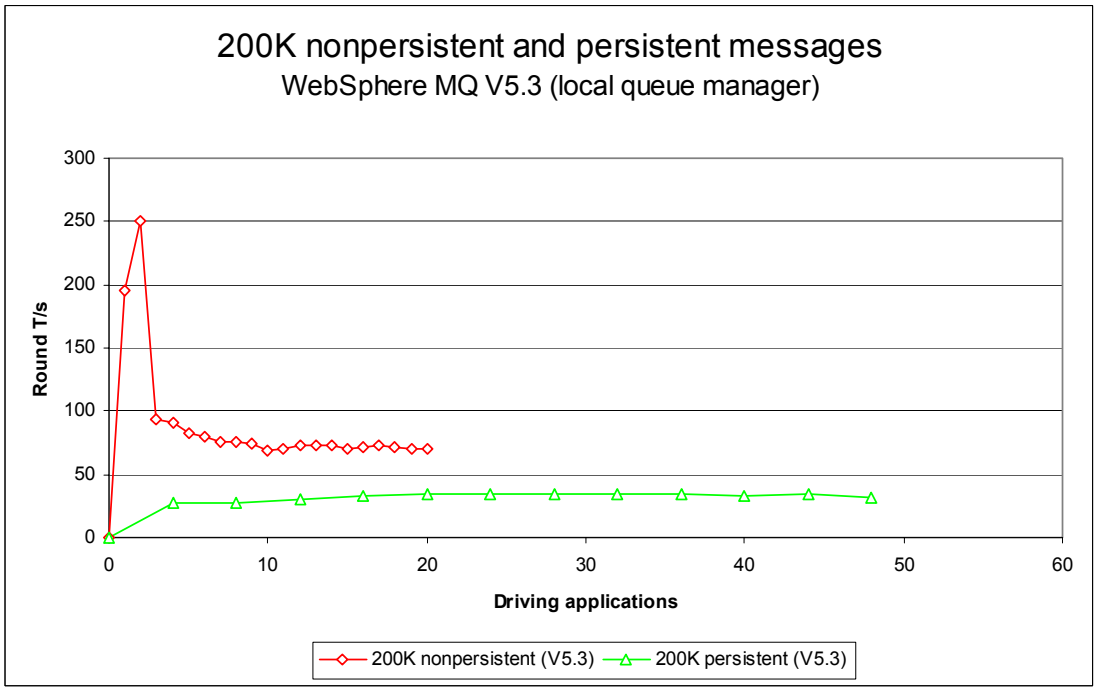

Figure 21 – 200 KB nonpersistent and persistent messages, local queue manager

Test name	Apps	Message size (KB)	Round trips/s	Response time (s)
local_np1	2	2	4557	0.001
local_np1_20K	2	20	2472	0.001
local_np1_200K	2	200	250	0.009
local_pm3	64	2	1014	0.071
local_pm3_20K	60	20	266	0.42
local_pm3_200K	16	200	33	0.546

Table 11 – 2, 20 and 200 KB messages, local queue manager

3.3 Large messages (20 and 200 KB) – client channels

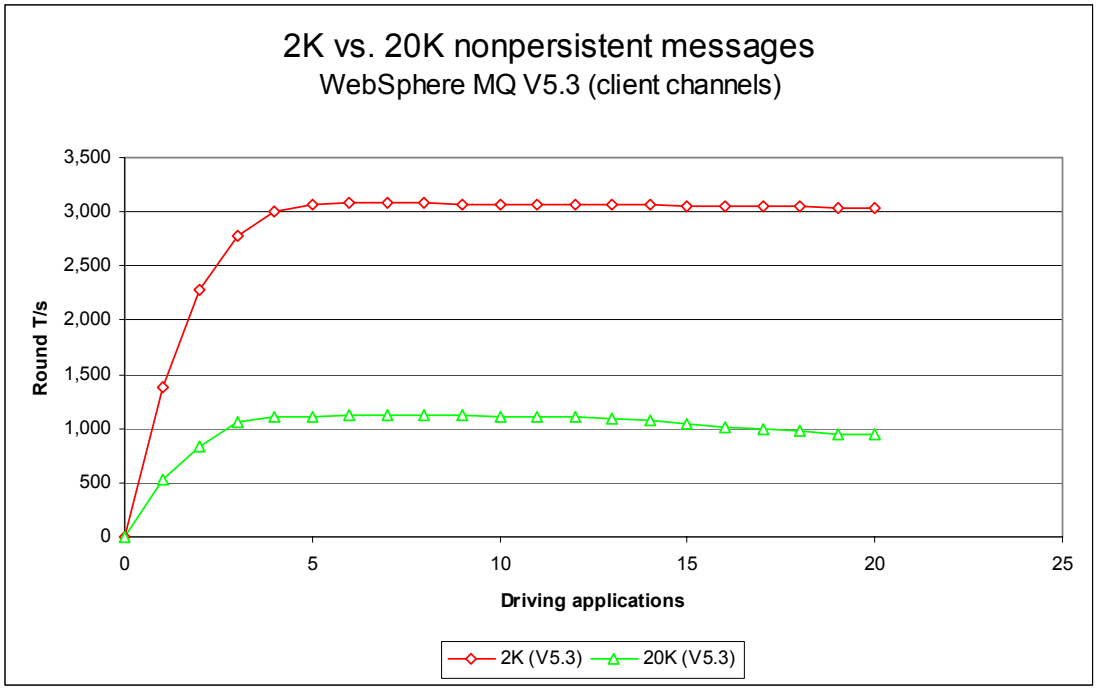

Figure 22 – 2 and 20 KB nonpersistent messages, client channels

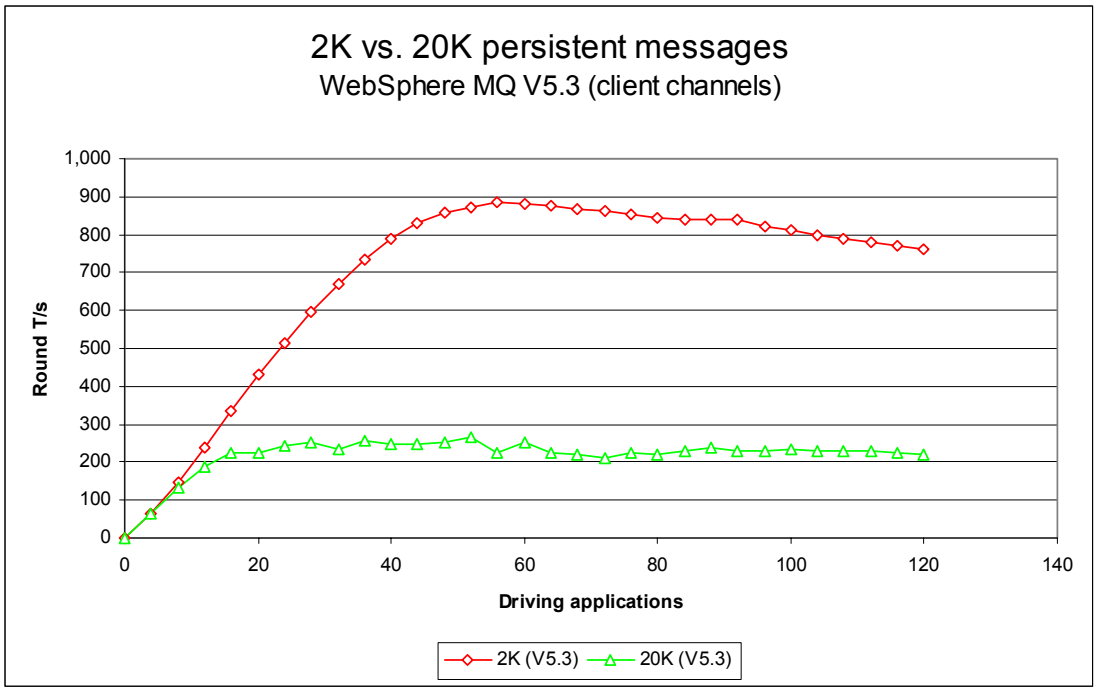

Figure 23 – 2 and 20 KB persistent messages, client channels

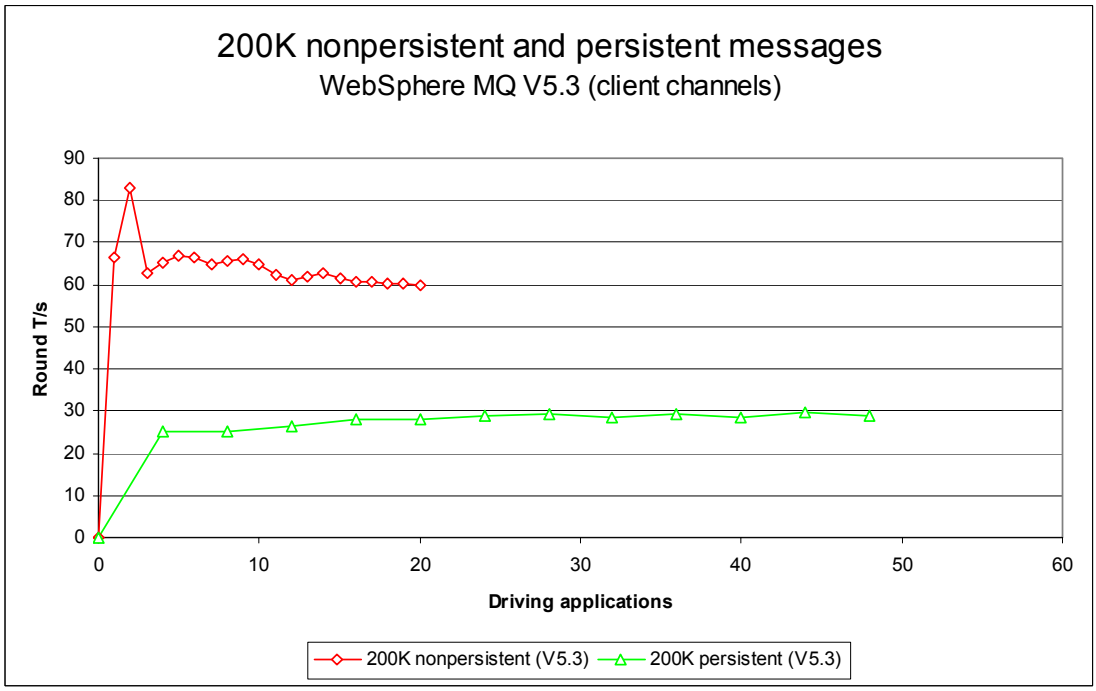

Figure 24 – 200 KB nonpersistent and persistent messages, client channels

Test name	Apps	Message size (KB)	Round trips/s	Response time (s)
clnp1	6	2	3084	0.002
clnp1_20K	7	20	1127	0.007
clnp1_200K	2	200	83	0.027
clpm3	56	2	886	0.070
clpm3_20K	52	20	266	0.273
clpm3_200K	16	200	28	0.644

Table 12 – 2, 20 and 200 KB messages, client channels

3.4 Large messages (20 and 200 KB) – distributed queuing

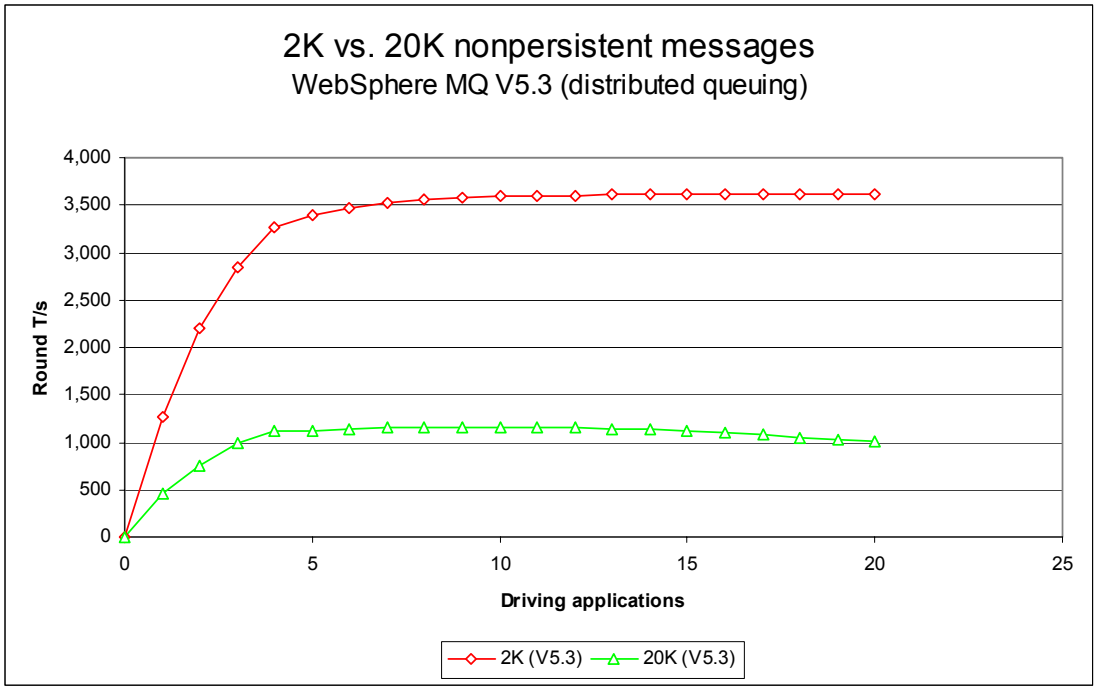

Figure 25 – 2 and 20 KB nonpersistent messages, server channels

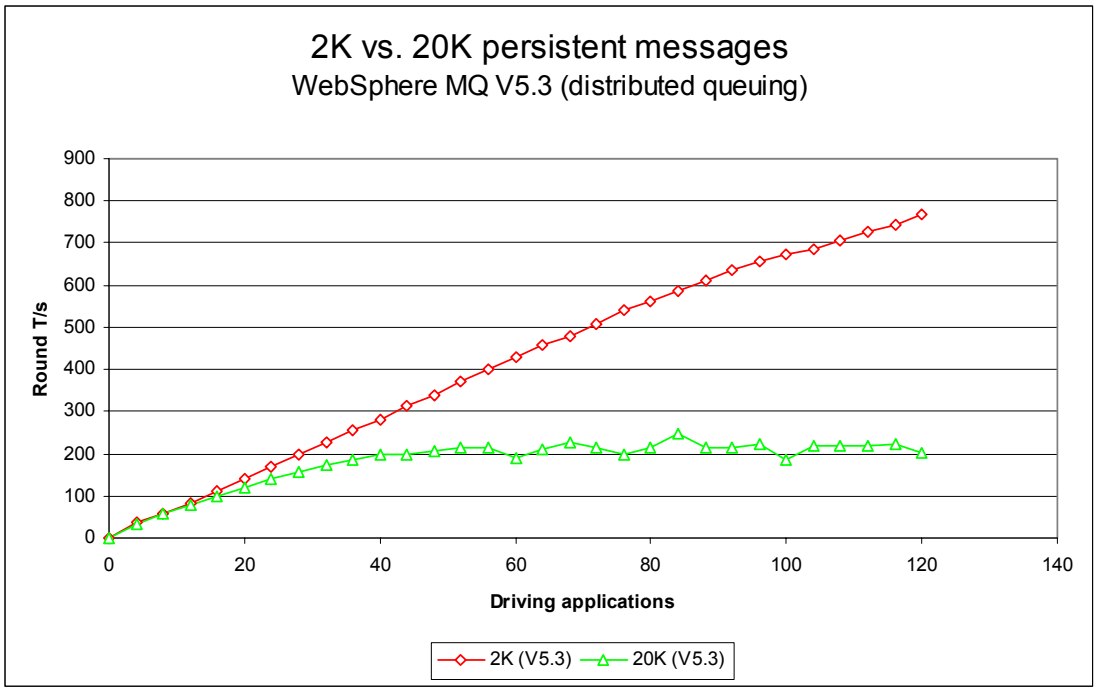

Figure 26 – 2 and 20 KB persistent messages, server channels

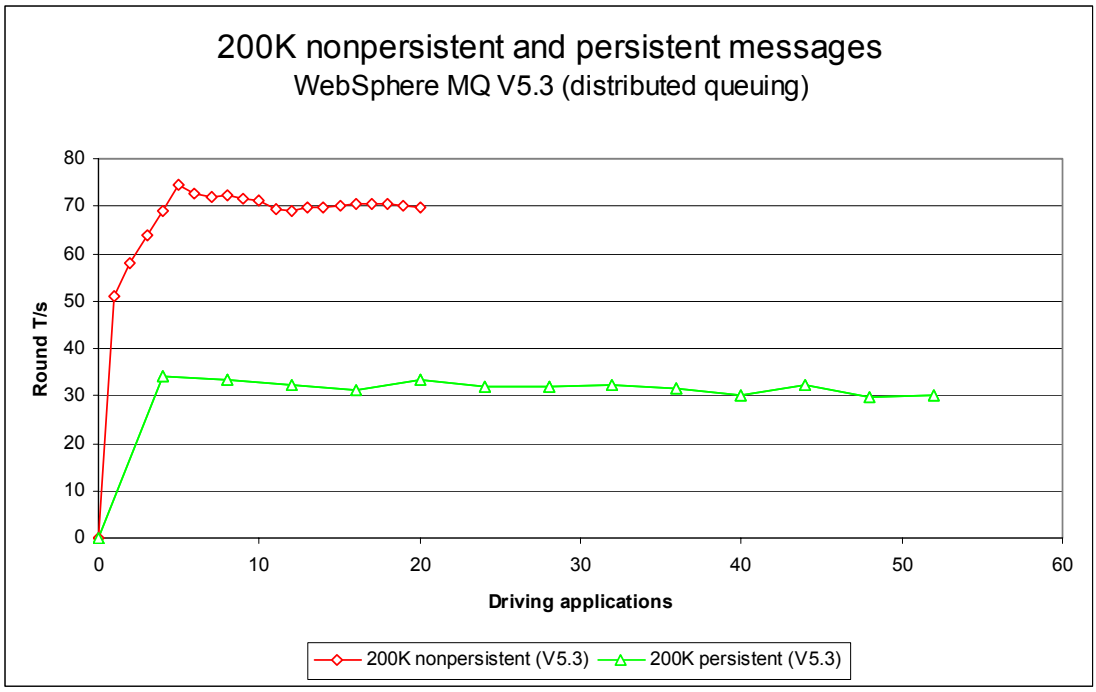

Figure 27 – 200 KB nonpersistent and persistent messages, server channels

Test name	Apps	Message size (KB)	Round trips/s	Response time (s)
dqnp1	18	2	3617	0.007
dqnp1_20K	9	20	1163	0.010
dqnp1_200K	5	200	75	0.076
dqpm1	120	2	767	0.180
dqpm1_20K	84	20	247	0.383
dqpm1_200K	8	200	23	0.391

Table 13 – 2, 20 and 200 KB messages, server channels

4 Trusted server application

Figure 28 below shows the improvement in throughput achieved for nonpersistent messages in the local queue manager scenario when the server application was run trusted ('fastpath').

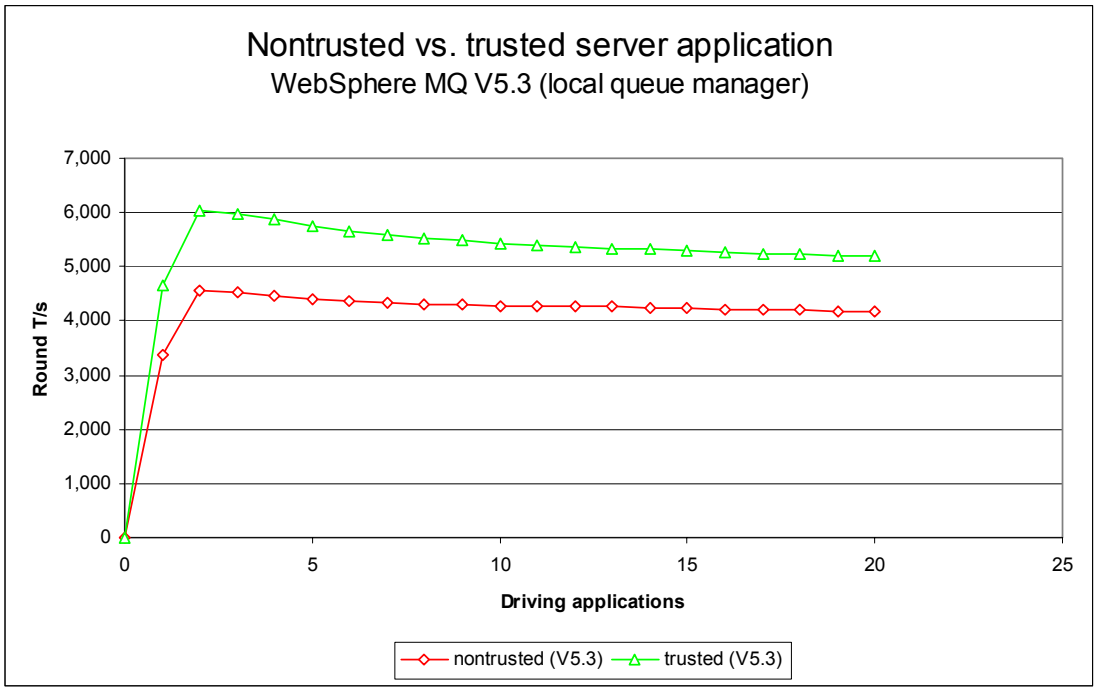

Figure 28 – Non-trusted vs trusted server application, local queue manager, 2KB nonpersistent messages

Test name	Scenario	Message type	Apps	Round trips/s	Response time (s)
local_np1_trusted	Local	Nonpersistent	2	6041	< 0.001
local_pm3_trusted	Local	Persistent	64	1104	0.067
clnp1_trusted	Client	Nonpersistent	7	3624	0.002
clpm3_trusted	Client	Persistent	60	960	0.075
dqnp1_trusted	Server	Nonpersistent	10	4075	0.003
dqpm1_trusted	Server	Persistent	208	956	0.264

Table 14 – Trusted server application, 2 KB messages, local queue manager, client channels and server channels

5 Short sessions

A short session describes a workload in which an MQI application processes only a few messages between connecting to and disconnecting from the queue manager. Triggered applications typically follow a short session profile. The measurements in this section were based on the following profile:

- connects to the queue manager,
- opens the common input queue and common reply queue,
- puts a request message to the common input queue, 5x
- gets the reply message from the common reply queue,
- closes both queues,
- disconnects from the queue manager.

As the number of connecting and disconnecting applications increases, the operating system and queue manager come under increasing load. While the connection and disconnection requests are being serviced, the queue manager has less time available to process messages. Therefore, fewer driving applications can be reconnected to the queue manager per second before the response time exceeds one second compared to a similar system in which the applications remain connected all the time. This is illustrated by comparing **Figure 29** below with **Figure 10.**

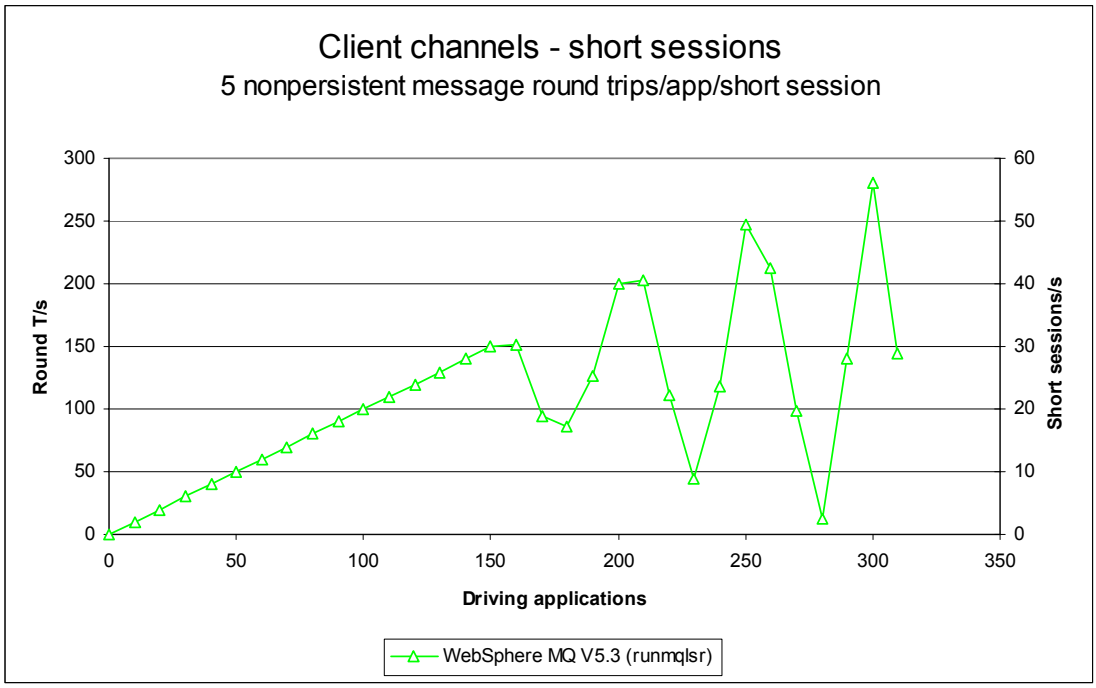

Figure 29 – Short sessions, client channels

Test name	Apps	Round trips/s	Short sessions/s	Response time (s)
clnp1_ss_r3600_runmqlsr	150	150	30	0.008

Table 15 – Short sessions, 2 KB nonpersistent messages, client channels

The table shows the peak throughput achieved before response times exceeded one second. Since there were five round trips per short session, the short session elapsed time would have approached five seconds when the round trip response time was approaching one second.

6 Performance and capacity limits

6.1 Client channels – capacity measurements

The measurements in this section illustrate the trade-off of workload (message rate) per client against the total number of MQI-clients connected to a single, remote queue manager.

Table 16 below shows the number of clients at which maximum throughput was achieved with zero think time vs. 1 message/app/s. All client connections were made trusted.

Test name	Apps	Rate /app/h	Round trips/s	Response time (s)
clnp1	6	n/a	3084	0.002
clnp1_r3600_runmqlsr	1100	3600	1098	0.020

Table 16 – Capacity measurements, 2 KB nonpersistent messages, client channels

Note 'Rate/app/h' refers to the number of messages put to the common input queue per driving application per hour. The 'clnp1' test used zero think time.

7 Performance tuning recommendations

This section summarises those tuning activities known to provide a significant performance benefit in WebSphere MQ V5.3. If applied inappropriately, some of the tuning recommendations described below may degrade the performance of a previously balanced system, especially if it already meets required performance criteria. The reader should closely monitor the result of tuning WebSphere MQ to be satisfied of no adverse effects.

7.1 Tuning the queue manager

7.1.1 Queue disk, log disk and message persistence

To avoid potential queue and log I/O contention due to the queue manager simultaneously updating a queue file and log extent on the same disk, it is important that queues and logs are located on *separate* and *dedicated* physical devices. Also, persistent messages should only be used if the message needs to survive a queue manager restart. In guaranteeing the recoverability of persistent messages, the pathlength through the queue manager is longer than for a nonpersistent message. However, cached disks may be used to minimise the time required to write a persistent message to the log.

7.1.1.1 Nonpersistent queue buffer

The default nonpersistent queue buffer size is 64 KB per queue. This can be increased to 1 MB using the *DefaultQBufferSize* parameter in the *TuningParameters* section of the registry. (Note: the *TuningParameters* section is not a documented external interface and may change from release to release. It is located under `HKEY_LOCAL_MACHINE\Software\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\<NameOfYourQueueManager>\`. The *DefaultQBufferSize* parameter must be specified in bytes. Remember to run *amqmdain regsec* at a command prompt to secure the registry for WebSphere MQ before restarting the queue manager.) The nonpersistent queue buffer is computationally less expensive because the queue manager is then less likely to use the file system to retrieve a message from the queue file. Increasing the queue buffer size allows the system to absorb peaks in message throughput at the expense of real storage. Defining queues using large nonpersistent queue buffers can degrade performance if the system is short of real memory.

Queues can be defined with different values of *DefaultQBufferSize*. If some queues need to be defined differently to others the values can be set in the *TuningParameters* section. When the queue manager is restarted existing queues will keep their earlier definitions and new queues will be created with the desired parameters. When a queue is opened resources are allocated according to the definition held on disk at the time the queue was created.

7.1.2 Log buffer size, log file size and number of log extents

To improve persistent message throughput *LogBufferPages* should be increased to its maximum configurable size of 512 x 4 KB pages = 2 MB; *LogFilePages* (i.e. `crtmqm -lf <LogFilePages>`) should be configured to a large size, for example: 16384 x 4 KB pages = 64 MB; and *LogPrimaryFiles* (i.e. `crtmqm -lp <LogPrimaryFiles>`) should be configured to a large number. The cumulative effect of this tuning will improve the throughput of the log buffer (permitting a maximum possible of 2 MB of log records to be written to the log disk in a single write), reduce the frequency of log switching (permitting a greater amount of log data to be written to one extent), and allow more time to prepare new linear logs or recycle old circular logs (especially important for long-running units of work). However, a large number of logs or a large log file size can result in the queue manager taking a long time to be created.

Changes to the queue manager *LogBufferPages* parameter take effect at the next queue manager restart. The log buffer size can be changed for all subsequent queue managers by changing the *LogBufferPages* parameter in the product's default *Log* section of the Windows registry.

It is unlikely that poor persistent message throughput will be attributed to the 2 MB limit of the log buffer size. It is possible to fill and empty the log buffer several times each second and reach a CPU limit writing data to the log buffer before a log disk bandwidth limit is reached.

7.1.3 Channels: standard or fastpath?

Fastpath channels and/or fastpath applications (see below for further discussion) can increase throughput for both nonpersistent and persistent messaging. For persistent messages the improvement is only for the path through the queue manager, and does not affect the performance of writing to the log disk. The reader should note that since the greater proportion of time in processing persistent messages is devoted to writing to the log disk, the performance improvement for fastpath channels is less apparent for persistent messages than for nonpersistent messages.

7.2 Tuning applications: design and configuration

7.2.1 Standard or fastpath?

The reader should be aware of the issues associated with writing and using fastpath applications—described in the *MQSeries Application Programming Guide*. Although customers are recommended to use fastpath channels, they are not recommended to use fastpath applications. If the performance gain offered by running fastpath is not achievable by other means it is essential that applications are rigorously tested running fastpath and never forcibly terminated (i.e. the application should always disconnect from the queue manager). Fastpath channels are documented in the *MQSeries Intercommunication Guide*.

7.2.2 Parallelism, batching, and triggering

An application should be designed wherever possible to run as multiple instances or with multiple threads of execution. Although the capacity of a multi-processor system can be fully utilised with a small number of applications using nonpersistent messages, more applications are required if the workload is mainly using persistent messages. Processing messages inside syncpoint can help reduce the amount of time the queue manager takes to write a group of persistent messages to the log disk. The behavior of a workload will also be subject to variability through cycles of light and heavy utilisation; therefore a degree of experimentation will be required to determine an optimum configuration.

Queue avoidance is a feature of the queue manager that allows messages to be passed directly from an ‘MQPUTer’ to an ‘MQGETer’ without the message being placed on a queue. This feature only applies to processing non-persistent messages outside of syncpoint. In addition to improving the performance of a workload with multiple parallel applications, the design should attempt to ensure that an application or application thread is always available to process messages on a queue (i.e. an MQGETer). Non-persistent messages outside of syncpoint then do not ever need to be physically placed on a queue.

Queue avoidance is less likely to be sustained as the MQPUTer applications increase in number. The reasons for this have a cumulative impact on performance. Consider, for example, the situation when nonpersistent messages are being placed on a queue quicker than they can be removed. The first effect is that messages begin to fill the nonpersistent queue buffer and MQGETers need to retrieve messages from the buffer rather than directly from an MQPUTer. A secondary effect is that as messages are spilled from the buffer to the queue disk, the MQGETers must wait for the queue manager to retrieve the message from the queue disk rather than from the queue buffer. While these problems can be reduced by arranging for more MQGETers, a performance degradation cannot necessarily be avoided.

Processing messages inside syncpoint (i.e. in batches) is more efficient than processing outside of syncpoint. As the number of messages in each batch increases the cost of processing each message decreases (while the total cost of the whole batch increases). Using syncpoint control is particularly true for persistent messaging as the queue manager

can write the entire batch of messages to the log disk in one go, whereas outside of syncpoint each message is written individually. However, inside of syncpoint, messages are not visible on the queue to other applications until the batch has been committed.

A triggered application typically follows the performance profile of a short session. It is advisable to make the disconnect interval an input parameter for the triggered application so as to facilitate performance-related adjustments in future. For example, in one production environment it may be more efficient for the application to remain connected to the queue manager between periods of message processing. However, in another environment it might be better for the triggered application to disconnect and terminate so as to reduce demand on the queue manager and operating system.

8 Measurement environment

8.1 Hardware

Unless otherwise stated one or two machines of the following specification were used for all of the tests described in this report:

Machine model	IBM Netfinity 8500R
Processor	Intel Pentium 3 Xeon 700 MHz, 2 MB L2 cache
Architecture	4-way SMP – but note that XP Professional only uses 2 of them
Memory (RAM)	8 GB – but note that XP Professional only uses 4GB maximum.
Disk	2 internal 10,000 rpm SCSI disks – 18 GB and 9 GB; 1 external 10,000 rpm SCSI disk – 9 GB
Network	1 Gigabit Ethernet

8.2 Software

Operating System	Microsoft Windows XP Professional
MQSeries/WebSphere MQ	WebSphere MQ for Windows, Version 5.3 CSD01
Compiler	Microsoft Visual C++ 6.0 Professional Edition

8.3 Workload description

8.3.1 MQI performance tool

The MQI tool is a suite of single-threaded applications which take it in turn to exercise a local queue manager by measuring elapsed time statistics for the eight main WebSphere MQ verbs: MQCONN(X), MQDISC, MQOPEN, MQCLOSE, MQPUT, MQGET, MQCMIT, and MQBACK. The queue manager is created using the command *crtmqm –lc –lf 2048*.

8.3.2 Scenario workload

Unless otherwise stated the queue manager's log was configured as follows for persistent message tests:

```
LogPrimaryFiles=4, LogSecondaryFiles=2, LogFileSize=4095,
LogBufferPages=512.
```

All the tests described in this report used circular logging and had MaxChannels set to 5000. MaxChannels refers to the Windows registry key under the 'Channels' section of the registry entry for the queue manager.

8.3.3 The driving application programs

The workload used simulated many driving applications running on a single driving machine. The applications were run trusted to conserve resources on the driving machine. This configuration is not typical of a customer environment and was only used to facilitate test coordination. Driving applications were multi-threaded with each thread performing a series of MQI calls. The number of threads in each application was adjusted according to whether the test was measuring a local queue manager scenario (**Figure 4**), a client channel scenario (**Figure 7**), or distributed queuing scenario (**Figure 12**). This was done to reduce storage overheads on the driving system. Each driving application thread performed the following sequence of actions:

- MQPUT of a request message to the common input queue.
- MQGET with indefinite wait to obtain a reply message from either a common reply queue (if using correlation ID) or from a unique reply queue corresponding to an individual driving application.

Unless otherwise stated the driving applications had zero think-time. This meant a driving application would send another request message as soon as it had received the reply to its previous request. For the 'rated' tests, however, each driving application was forced to wait until a specified time had elapsed since it had put the previous request message before it would send the next request message. In both the zero think-time tests and the rated tests a driving application would never send another request message until it had received the reply to its previous request message.

For the client and distributed scenarios the channels were run trusted (there were no channels in the local queue manager scenario by definition).

Message size: For the release highlights and performance headlines (including rated messaging tests), a 2K message size was used. For the large message measurements a 20K and 200K message size was used.

Message rate: In all but the *rated* and *capacity limit* tests, message processing was performed in a *tight-loop*. In the *rated* tests, a message rate of 1 round trip per driving application per *second* was used.

Nonpersistent and persistent messages were used in tests.

Note: the driving applications gathered timing information for all MQI calls using a high-resolution timer.

8.3.4 The server application program

The server application is a multi-threaded program that was configured to use 5 threads for processing nonpersistent messages, and 20 or 40 threads for processing persistent messages. Each server thread performed the following sequence of MQI calls:

- MQGET with indefinite wait to retrieve a request message from the common input queue,
- MQPUT to the common reply queue or, if correlation ID is used, to a unique reply queue per driving application.

Nonpersistent messaging was done outside of syncpoint control. Persistent messaging was done inside of syncpoint control. The average message throughput expressed as a number of round trips per second was calculated and reported by the server program. Unless otherwise stated the server program was run non-trusted.

8.3.5 Test Descriptions

Local_np: A local test with both driver and server applications running on the same hardware. The test uses non-persistent messages and unless otherwise stated tests are run with up to 20 client connections. The aim of this test is to get the highest possible throughput without network or channel restrictions.

Local_pm: A local test with both driver and server applications running on the same hardware. The test uses persistent messages and unless otherwise stated tests are designed to run to 120 client connections. The aim of this test is to get the highest possible throughput without network or channel restrictions.

Queue manager log configuration:

```
LogPrimaryFiles=4, LogFilePages=4095, LogBufferPages=512
```

Clnp A test where driving applications connect using MQI-client channels to a remote queue manager on separate physical hardware, the server application is still local to the queue manager. The test uses non-persistent messages and unless otherwise stated tests are designed to run to 20 client connections. The aim is to provide measurements for client applications (a common set-up for customers).

Clpm A test where driving applications connect using MQI-client channels to a remote queue manager on separate physical hardware, the server application is still local to the queue manager. The test uses persistent messages and unless otherwise stated tests are designed to run to 120 client connections. The aim is to provide measurements for client applications (a common set-up for customers).

Queue manager log configuration:

```
LogPrimaryFiles=4, LogFilePages=4095, LogBufferPages=512
```

Dqnp A distributed test, the driving applications connect to a local queue manager and the server to another local queue on different hardware, the two queue managers then communicate via server channels. The test uses non-persistent messages and unless otherwise stated tests are designed to run to 20 driving applications. The aim of this test is to provide measurement of the server channels and comparison with the client and local tests.

Dqpm A local distributed test, the driving applications connect to a local queue manager and the server to another local queue on different hardware, the two queue managers then communicate via server channels. The test uses persistent messages and unless otherwise stated tests are designed to run to 120 client connections. The aim of this test is to provide measurement of the server channels and comparison to client and local tests. The queue manager log configuration is:

```
LogPrimaryFiles=4, LogFilePages=4095, LogBufferPages=512
```

MQPUT + MQGET

For this test a simple application records the time taken to do the MQ verbs MQPUT and MQGET, each verb is repeated 5000 times on an empty queue and the 90th percentile is shown in the graph. The tests are repeated for both persistent and non-persistent messages. Also, both run trusted and nontrusted. Queue manager log configuration:

```
LogPrimaryFiles=3, LogFilePages=2048
```

Trusted server

The trusted tests are run with the same settings and scenarios as the other tests except the server application now connects directly to the queue manager rather then through an agent process. This should give notable performance benefits. However it is not recommended for general usage.

Configuration for Large Message Tests

The queue manager's log was configured in the usual way for the 2 and 20 KB persistent message tests (i.e., LogPrimaryFiles=4, LogSecondaryFiles=2, LogFileSize=4095, LogBufferPages=512). However, for all the 200 KB persistent message tests the queue manager's log was configured to use more log files to cope with the larger messages: LogPrimaryFiles=12, LogSecondaryFiles=3, LogFileSize=16384, LogBufferPages=512.

9 Glossary

Apps	The number of driving applications connected to the queue manager at the point where the performance measurement is given
Rate/App/h	The intended message throughput rate (round trips per hour) for each of the driving applications. In practice the system only achieved this throughput rate whilst it was not constrained.
Resp time (s)	The average response time each round trip, as measured and averaged by all the driving applications
Response time (s)	The average duration in seconds for each round trip, as measured and averaged by all the driving applications.
Round T/s or Rounds trips/s	The average achieved message throughput rate (request messages per second) of all the driving applications together, measured by the server application.
Test name	The name of the test Note: the test names in some cases are rather long. This is done to provide a descriptive qualification of the test measurement to relate to the performance discussion in the sections throughout the document: **local** => local queue manager test scenario **cl** => client channel test scenario **dq** => distributed queuing test scenario **np** => nonpersistent messages **pm** => persistent messages **r3600** => 1 round trip per driving application per second
Zero think-time	The driving application sent a new request message as soon as it had received the reply to its previous request message.