WebSphere MQ for Windows V5.3 - Performance Evaluations on Windows 2000 Advanced Server

4 November 2002

Lucas Partridge

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

Take Note!

Before using this report be sure to read the general information under “Notices”.

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

First edition - July 2002.
Second edition - August 2002.
Third edition - November 2002.
This edition applies to V1.2 of WebSphere MQ for Windows V5.3 – Performance Evaluations and to all subsequent releases and modifications until otherwise indicated in new editions.

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 2000 Advanced Server was used as the device under test for all the measurements in this report. For full details of the measurement environment see page 35.

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.

Readers should have a general awareness of the Windows 2000 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 Alexander Russell and Richard Eures for their help in producing this report.

How this document is arranged

Release highlights

Page: 1

Outlines the performance improvements achieved in WebSphere MQ V5.3 compared to MQSeries V5.2. 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 server channel pairs that could be connected between two queue managers, using a fixed think time between messages, until the response time exceeded one second.

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: 26

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

Short sessions

Page: 27

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.

Performance and capacity limits

Page: 29

Shows:

- how many MQI-client channels were connected into a single queue manager, with a server application processing one nonpersistent round trip per MQI-client per minute.

- how many server channel pairs were connected between two queue managers on separate machines, with a server application on one of the machines processing one nonpersistent round trip per server channel pair per minute.

Performance tuning recommendations

Page: 32

Provides advice on how to design applications and tune the queue manager and operating system to achieve maximum performance benefits from a WebSphere MQ system.

Measurement environment

Page: 35

Describes the hardware and software environment and the workload scenarios used to produce the results in this report.

Glossary

Page: 38

Explains the terms used in the tables and elsewhere in this report.

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 Improvements to nonpersistent and persistent messaging

Compared to MQSeries for Windows NT and Windows 2000 V5.2, WebSphere MQ for Windows V5.3 showed the following performance improvements:

- Peak nonpersistent message throughput has increased by 28% in a local queue manager environment, 24% in an MQI-client environment, and 26% in a distributed queuing environment.
- Peak persistent message throughput has increased by 266% in a local queue manager environment, 210% in an MQI-client environment, and 87% 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 on WebSphere MQ V5.3 and MQSeries V5.2.

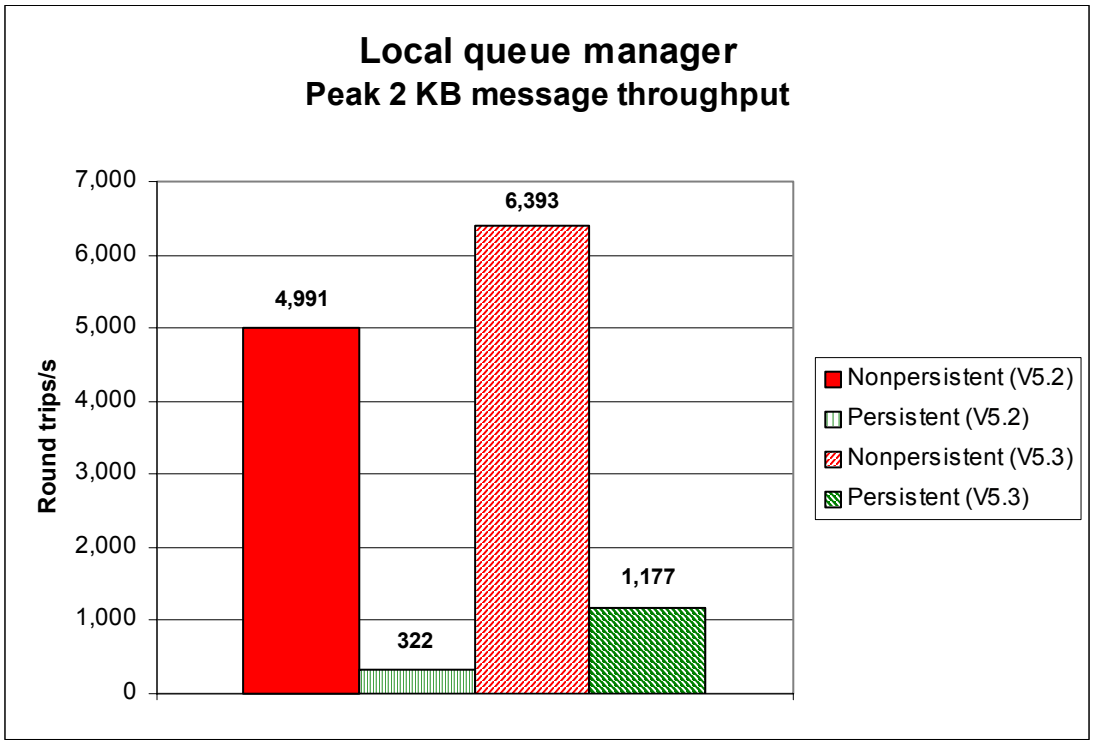

Figure 1 – Peak 2 KB message throughput, local queue manager

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 on WebSphere MQ V5.3 and MQSeries V5.2.

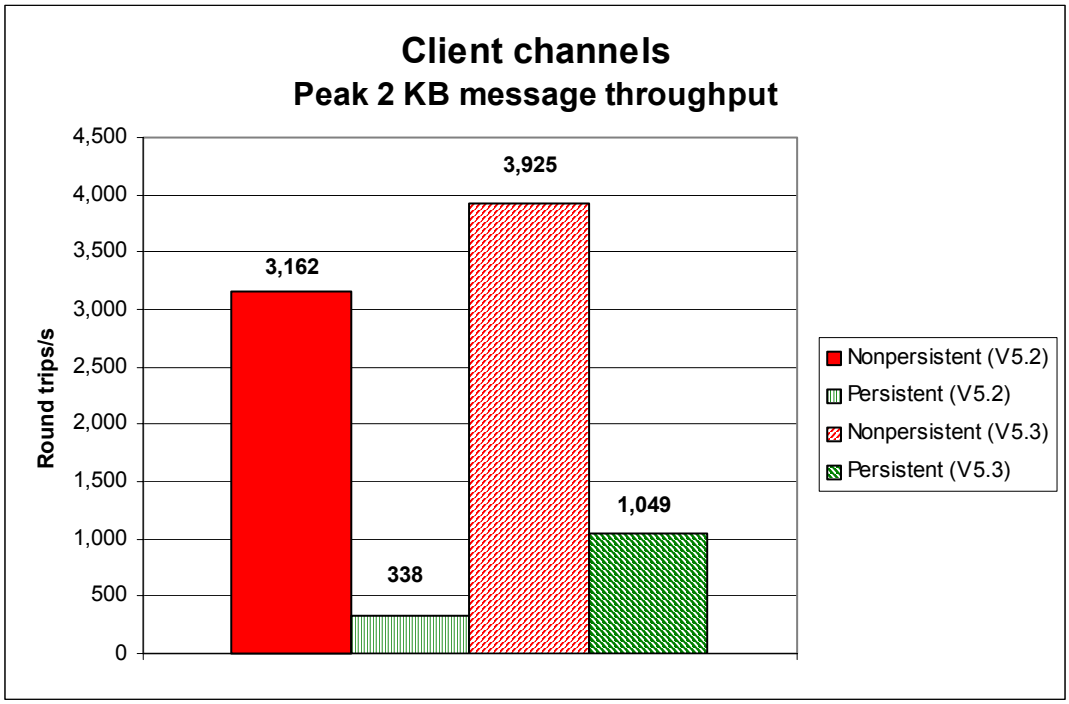

Figure 2 – Peak 2 KB message throughput, client channels

1.4 Distributed queuing – peak message throughput

Figure 3 below shows the peak message throughput achieved for nonpersistent and persistent messages with server channels on WebSphere MQ V5.3 and MQSeries V5.2. Note that throughput was still increasing with the number of driving applications for all but the nonpersistent WebSphere MQ V5.3 measurement so caution is advised in interpreting the peak figures shown (see also **Figure 13** and **Figure 14**).

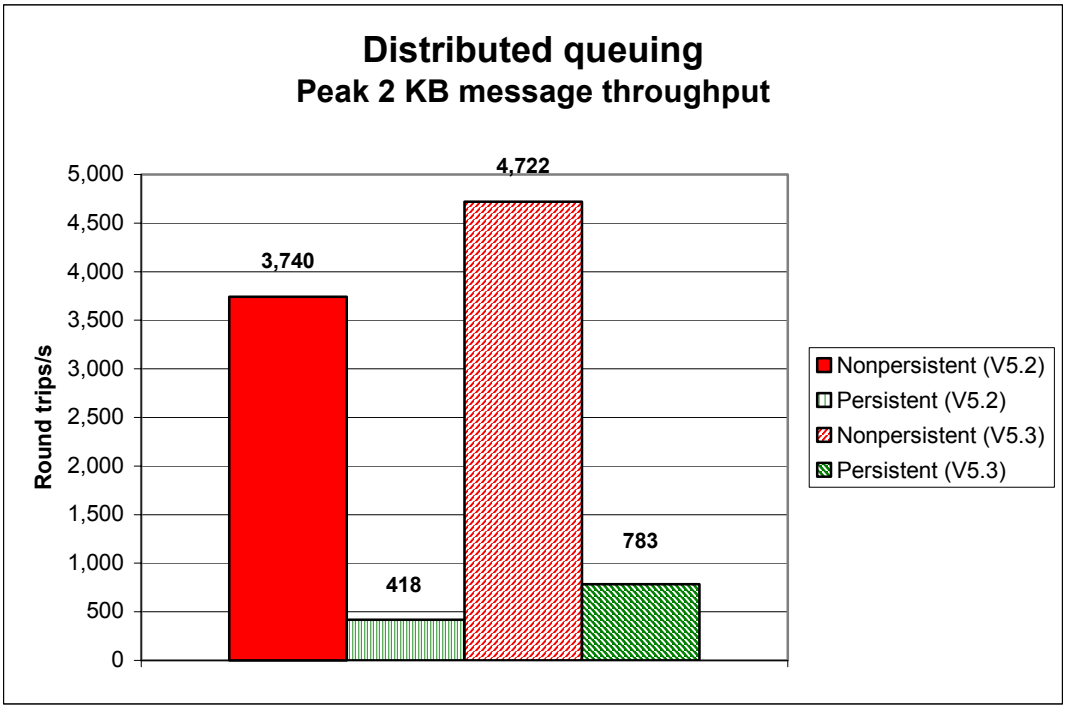

Figure 3 – Peak 2 KB message throughput, distributed queuing

1.5 Capacity limits

- Fastpath client (MQI) channel connection limits: 11,800 using one reply queue per client, or 13,000 using one reply queue for all the clients.
- Distributed (server) channel connection limits: 6,000 (and not constrained).

2 Performance headlines

2.1 Local queue manager scenario

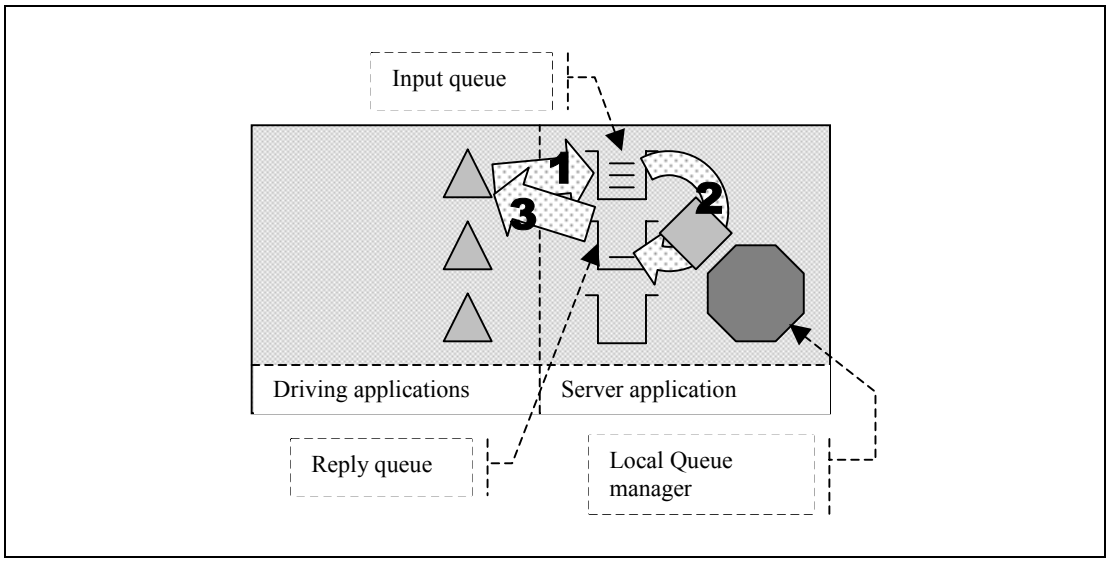

Figure 4 – Connections into a local queue manager

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** below show the peak nonpersistent and persistent message throughputs achieved on WebSphere MQ V5.3 and MQSeries V5.2 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.

2.1.1 Nonpersistent messages – local queue manager

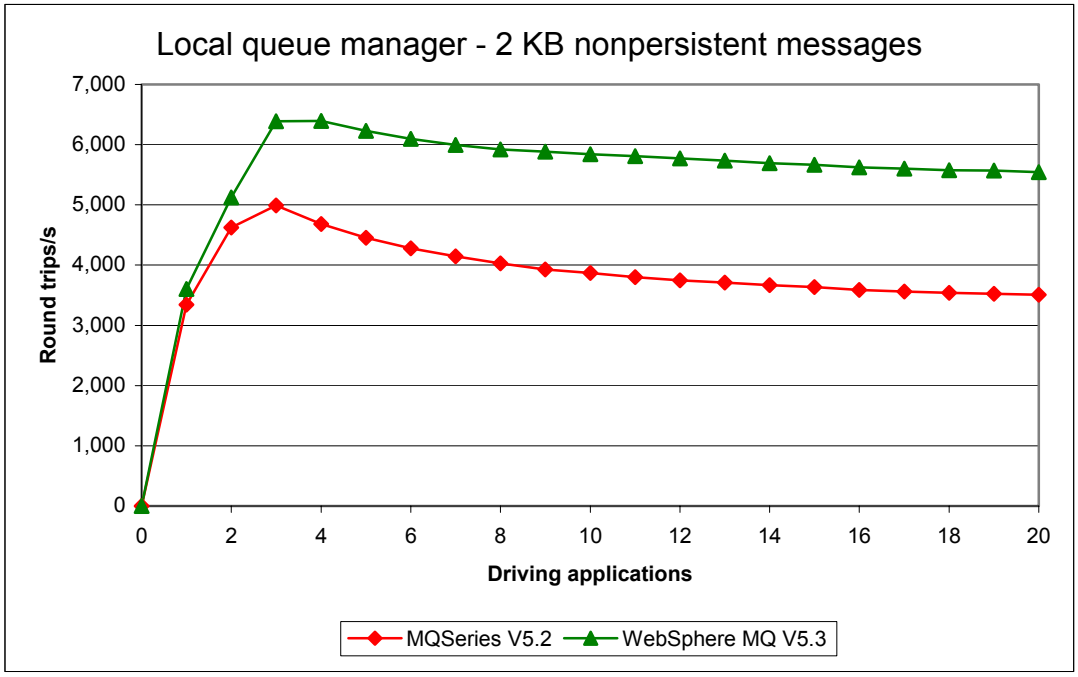

Figure 5 – Performance headline, nonpersistent messages, local queue manager

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	**3** (4) (20)	**4,991** (4,684) (3,506)	n/a	0.001 (0.001) (0.007)
WebSphere MQ V5.3	(3) **4** (20)	(6,389) **6,393** (5,542)	(+26) +38 (+58)	(0.001) 0.001 (0.004)

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

Note: The figures in bold show the peak throughput obtained for MQSeries V5.2 and WebSphere MQ V5.3 respectively. The figures in parentheses allow direct comparison of one version of the product with the other at the same number of driving applications. Refer to the **Glossary** for a description of each of the table column headings.

2.1.2 Persistent messages – local queue manager

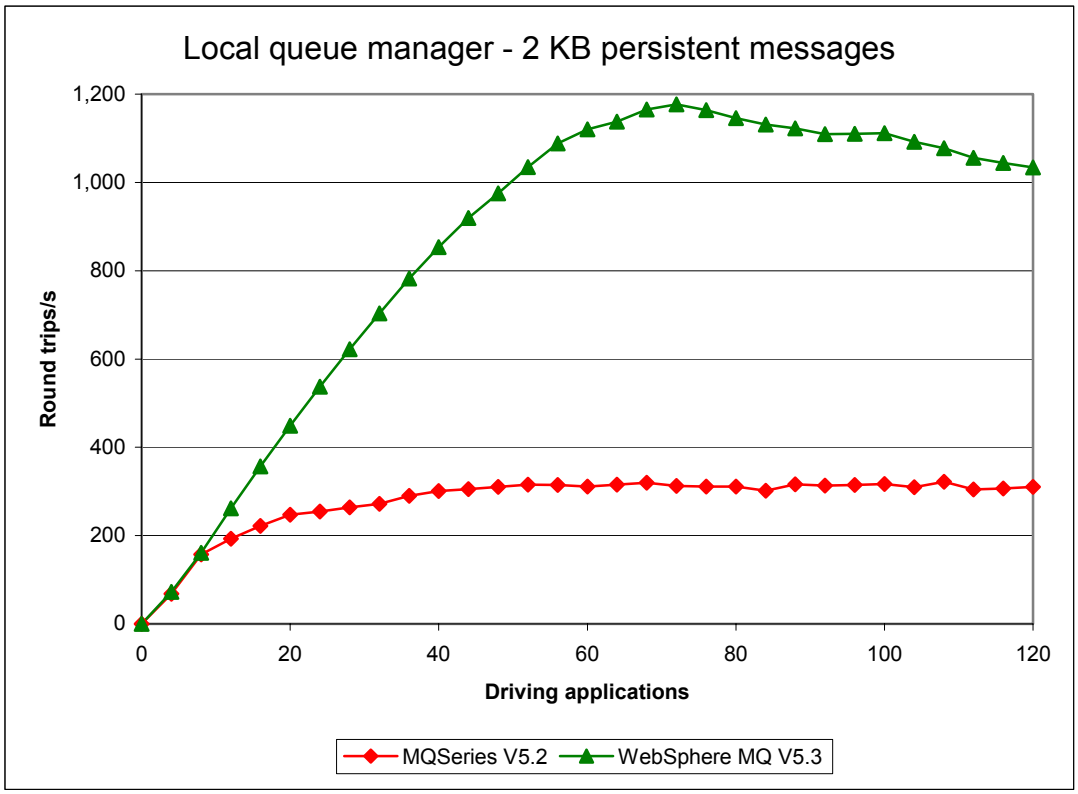

Figure 6 – Performance headline, persistent messages, local queue manager

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	(72) **108** (120)	(312) **322** (310)	n/a	(0.355) 0.419 (0.427)
WebSphere MQ V5.3	**72** (108) (120)	**1,177** (1,077) (1,034)	+277 (+234) (+234)	0.072 (0.115) (0.135)

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

2.2 Client channels scenario

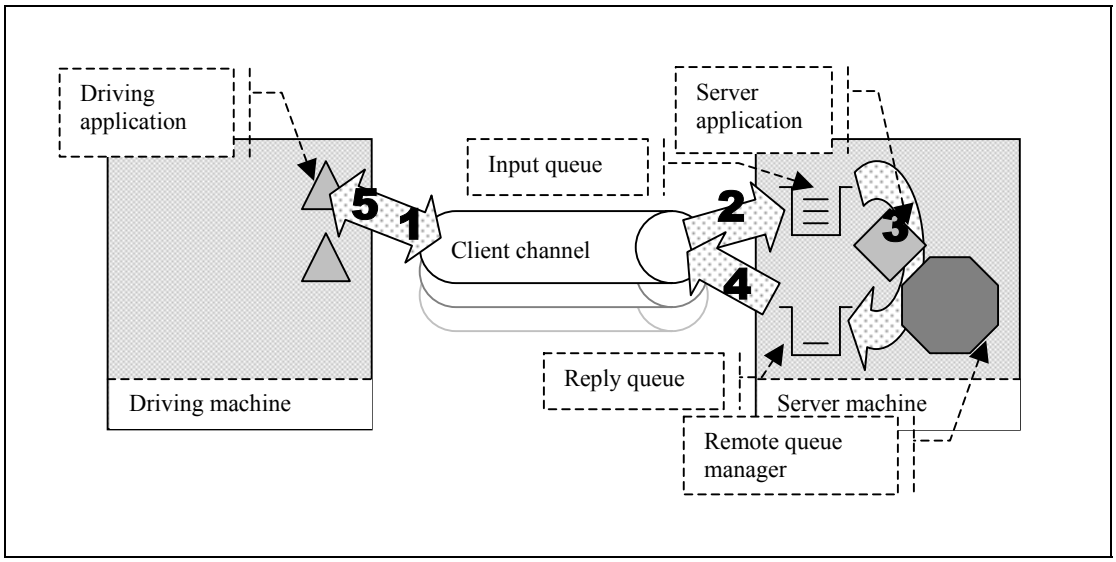

Figure 7 – MQI-client channels into a remote queue manager

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 on WebSphere MQ V5.3 and MQSeries V5.2 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

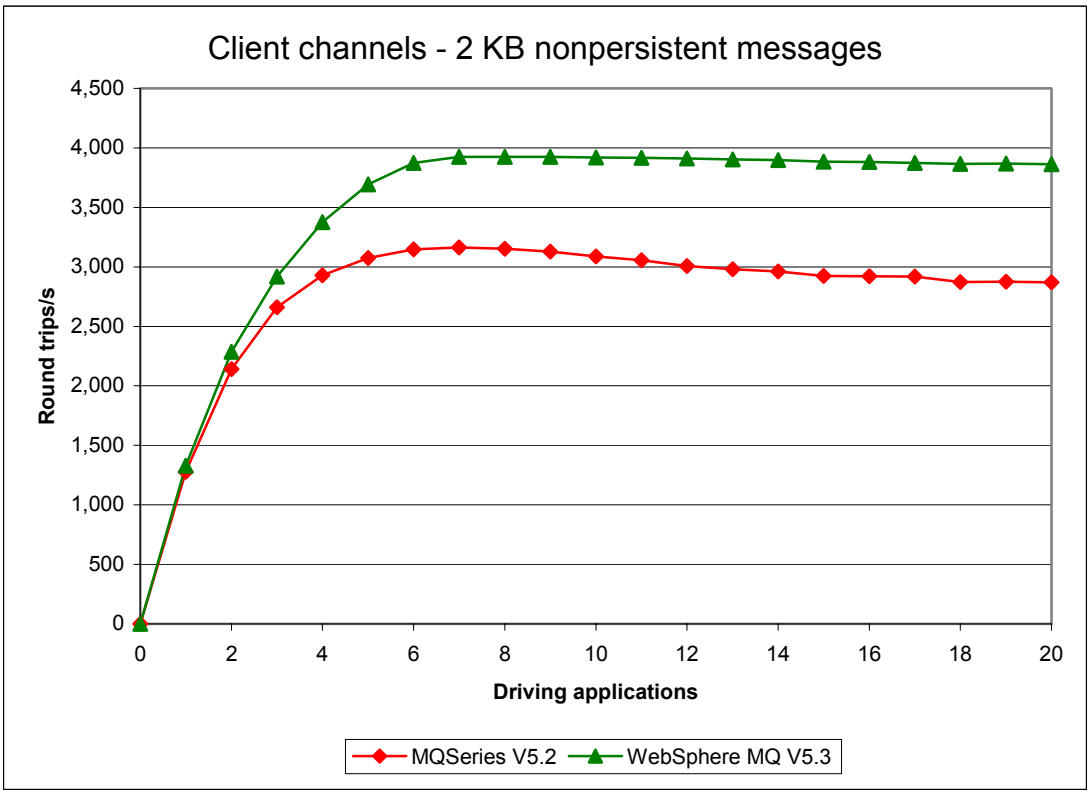

Figure 8 – Performance headline, nonpersistent messages, client channels

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	**7** (9) (20)	**3,162** (3,128) (2,871)	n/a	0.003 (0.003) (0.008)
WebSphere MQ V5.3	(7) **9** (20)	(3,925) **3,925** (3,862)	(+24) +25 (+35)	(0.002) 0.003 (0.007)

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

2.2.2 Persistent messages – client channels

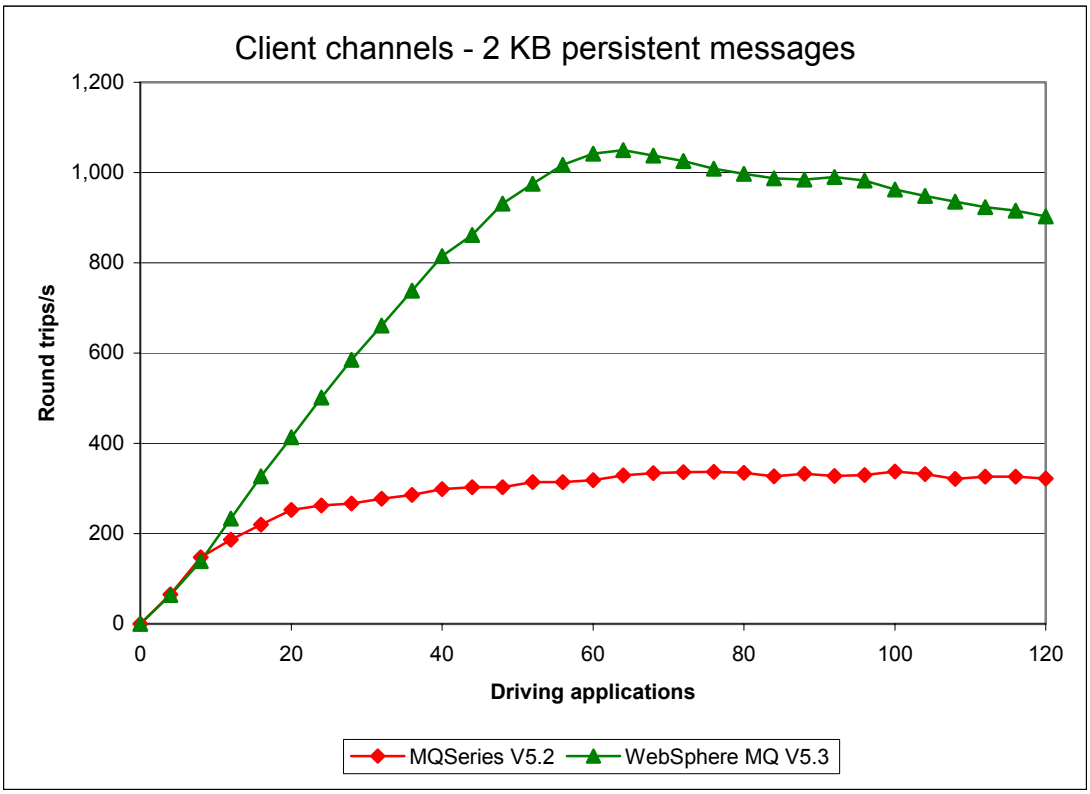

Figure 9 – Performance headline, persistent messages, client channels

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	(64) **100** (120)	(329) **338** (322)	n/a	(0.207) 0.351 (0.419)
WebSphere MQ V5.3	**64** (100) (120)	**1,049** (962) (903)	+219 (+185) (+180)	0.074 (0.124) (0.154)

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

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.

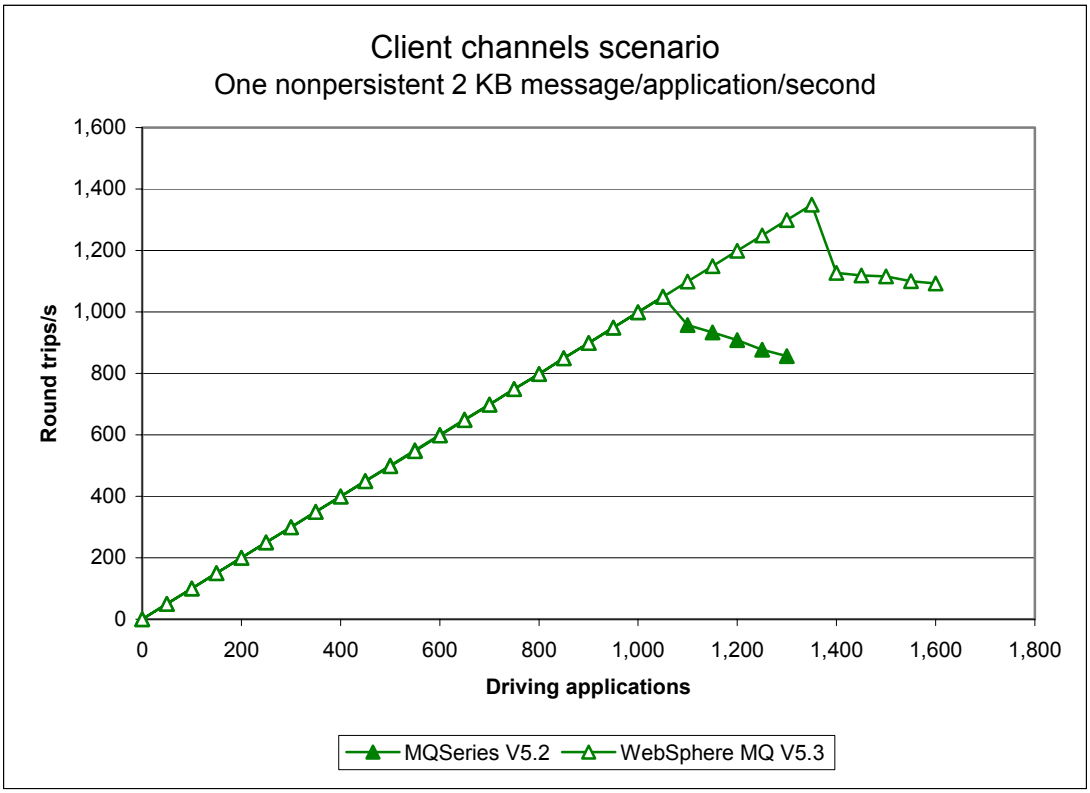

Figure 10 – Rated test, nonpersistent messages, client channels

Figure 10 above shows that WebSphere MQ V5.3 can support more client connections than MQSeries V5.2 when using nonpersistent messages (1,350 versus 1,050 clients respectively before response time exceeded one second).

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	1,050	1,049	n/a	0.039
WebSphere MQ V5.3	1,350	1,349	+29	0.010

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

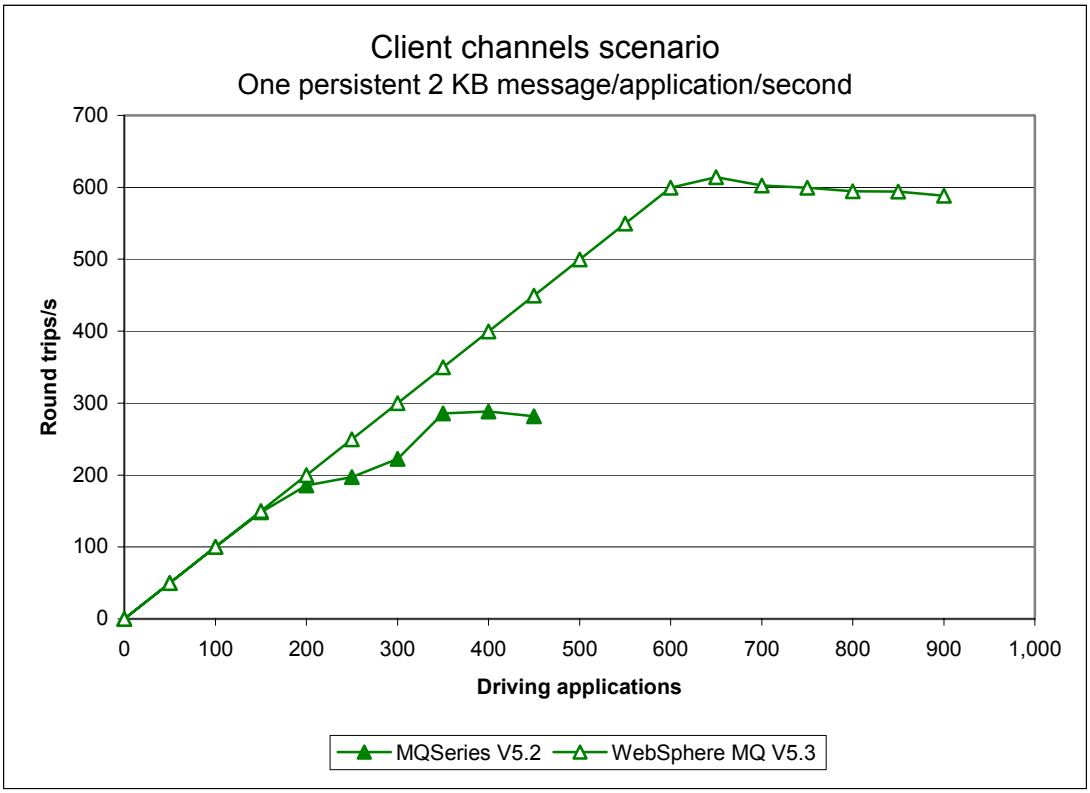

Figure 11 – Rated test, persistent messages, client channels

Figure 11 above shows that WebSphere MQ V5.3 can support more client connections than MQSeries V5.2 when using persistent messages (650 versus 300 clients respectively before response time exceeded one second).

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	300	223	n/a	0.532
WebSphere MQ V5.3	650	614	+175	0.802

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

2.3 Distributed queuing scenario

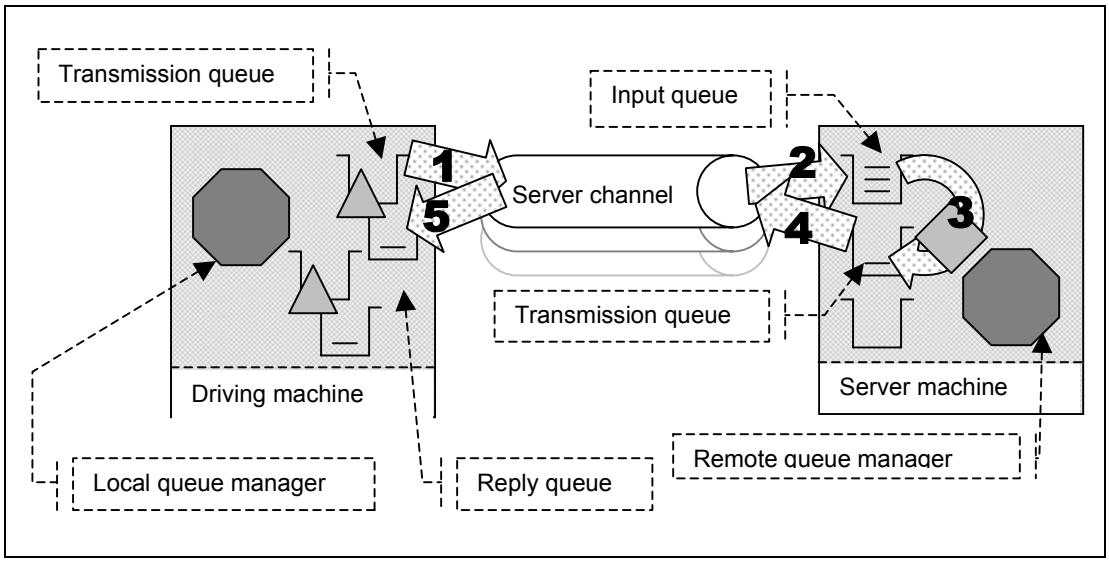

Figure 12 – Server channels between two queue managers

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 on WebSphere MQ V5.3 and MQSeries V5.2 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

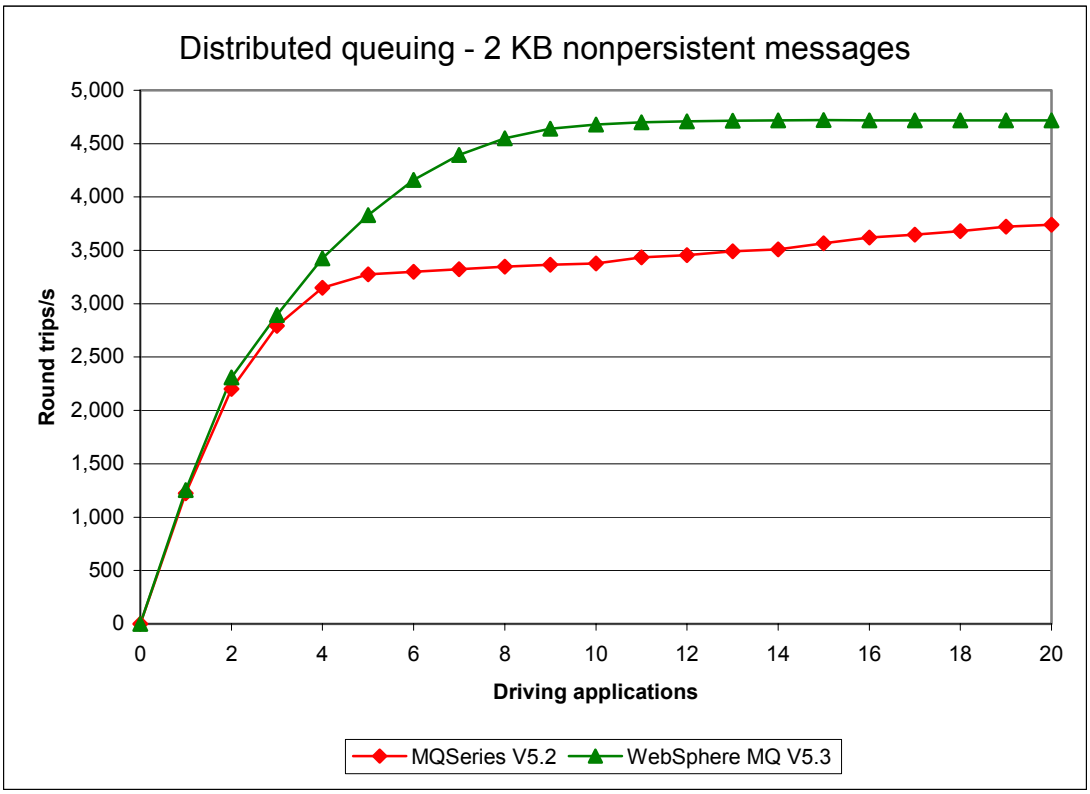

Figure 13 – Performance headline, nonpersistent messages, server channels

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	(15) **20**	(3,565) **3,740**	n/a	(0.006) 0.007
WebSphere MQ V5.3	**15** (20)	**4,722** (4,717)	+32 (+26)	0.003 (0.004)

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

Note that throughput was still increasing with the number of driving applications for MQSeries V5.2 when the test was completed at 20 applications.

2.3.2 Persistent messages – server channels

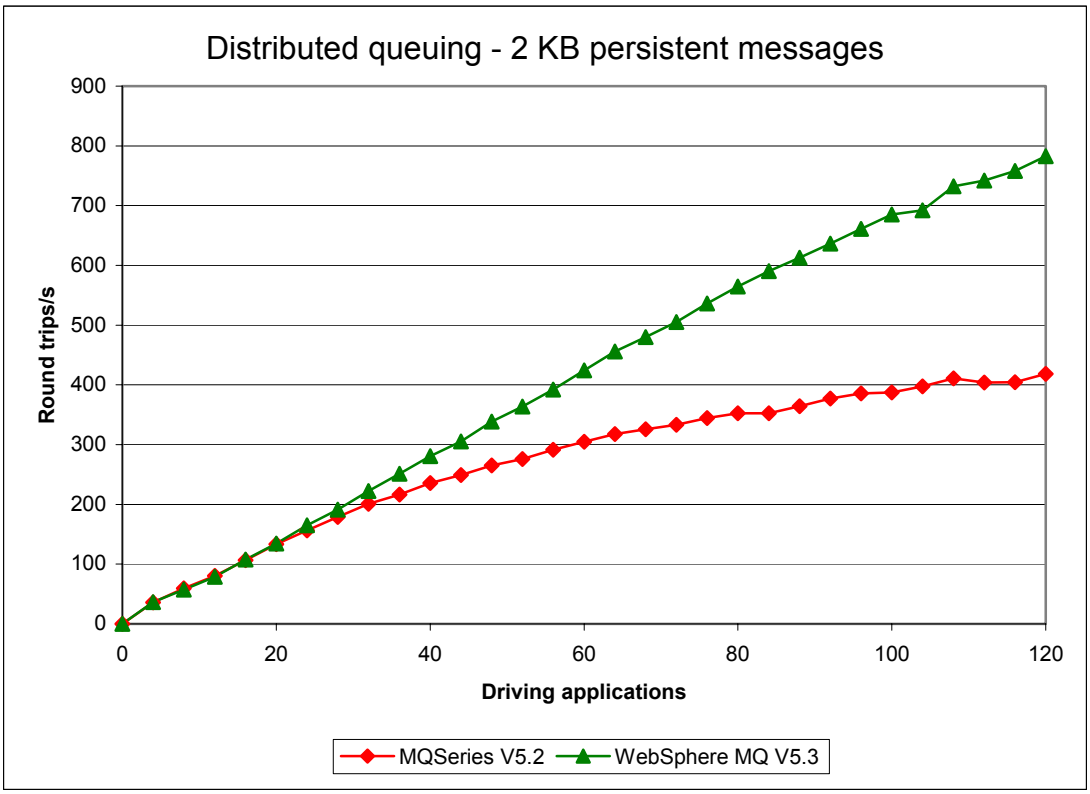

Figure 14 – Performance headline, persistent messages, server channels

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	120	418	n/a	0.346
WebSphere MQ V5.3	120	783	+87	0.177

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.

2.3.3 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.

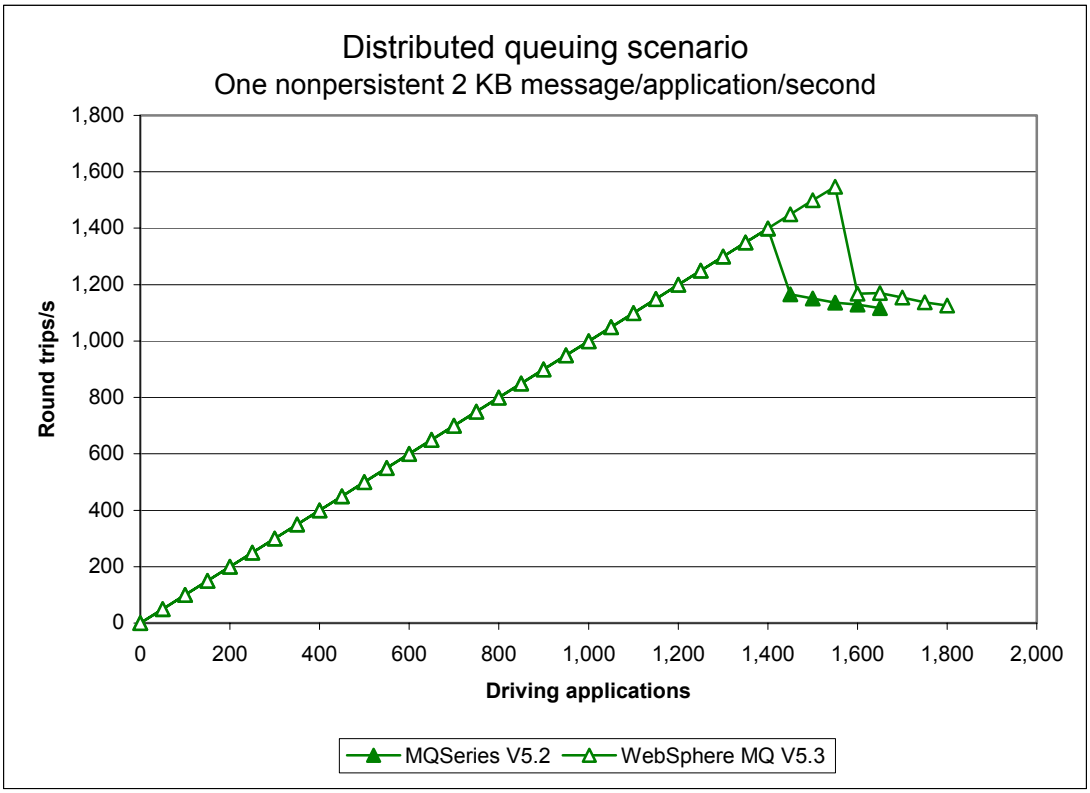

Figure 15 – Rated test, nonpersistent messages, server channels

Figure 15 above shows that WebSphere MQ V5.3 can support more driving applications that are connecting over a fixed number of server channels than MQSeries V5.2 when using nonpersistent messages (1,550 versus 1,400 driving applications respectively before response time exceeded one second).

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	1,400	1,398	n/a	0.027
WebSphere MQ V5.3	1,550	1,546	+11	0.027

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

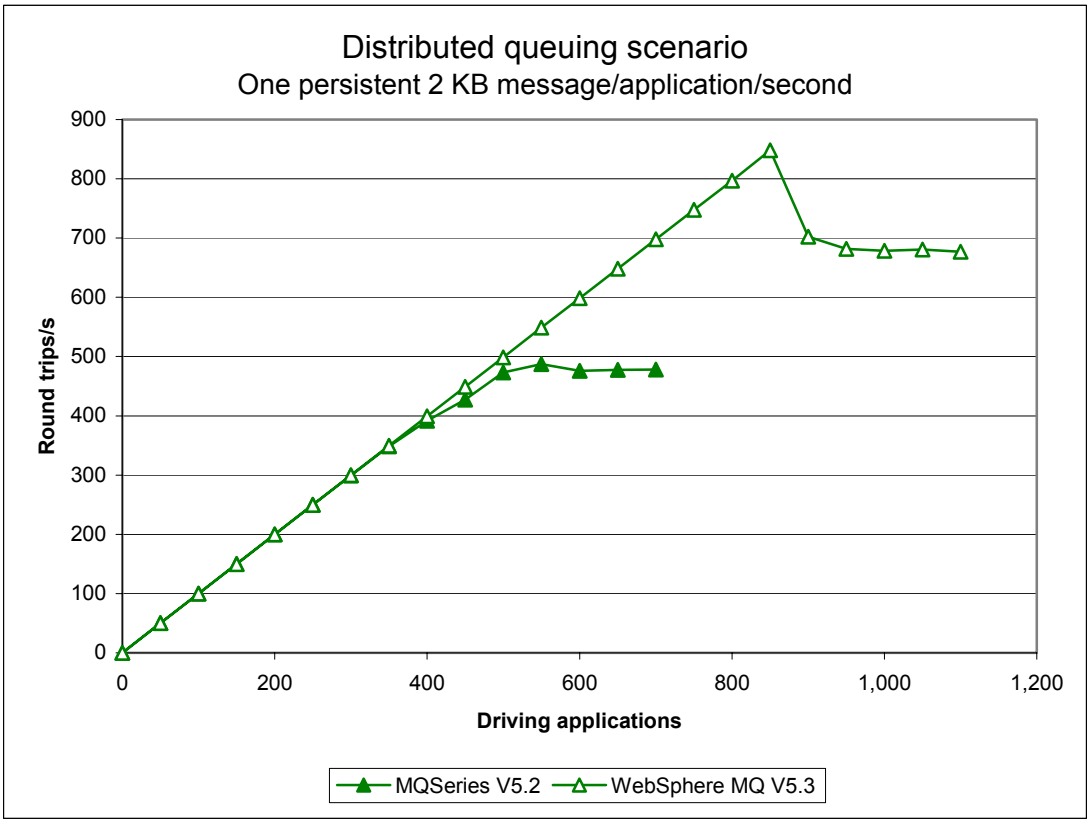

Figure 16 – Rated test, persistent messages, server channels

Figure 16 above shows that WebSphere MQ V5.3 can support more driving applications that are connecting over a fixed number of server channels than MQSeries V5.2 when using persistent messages (850 versus 500 driving applications respectively before response time exceeded one second).

Product version	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
MQSeries V5.2	500	473	n/a	0.934
WebSphere MQ V5.3	850	848	+79	0.212

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

3 Large messages

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

Figure 17 to **Figure 20** show that WebSphere MQ V5.3 has faster response times for MQPUT/MQGET pairs for all nonpersistent and persistent message sizes between 50 bytes and 2 MB. 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 256 KB and then for 50 bytes to 2 MB.

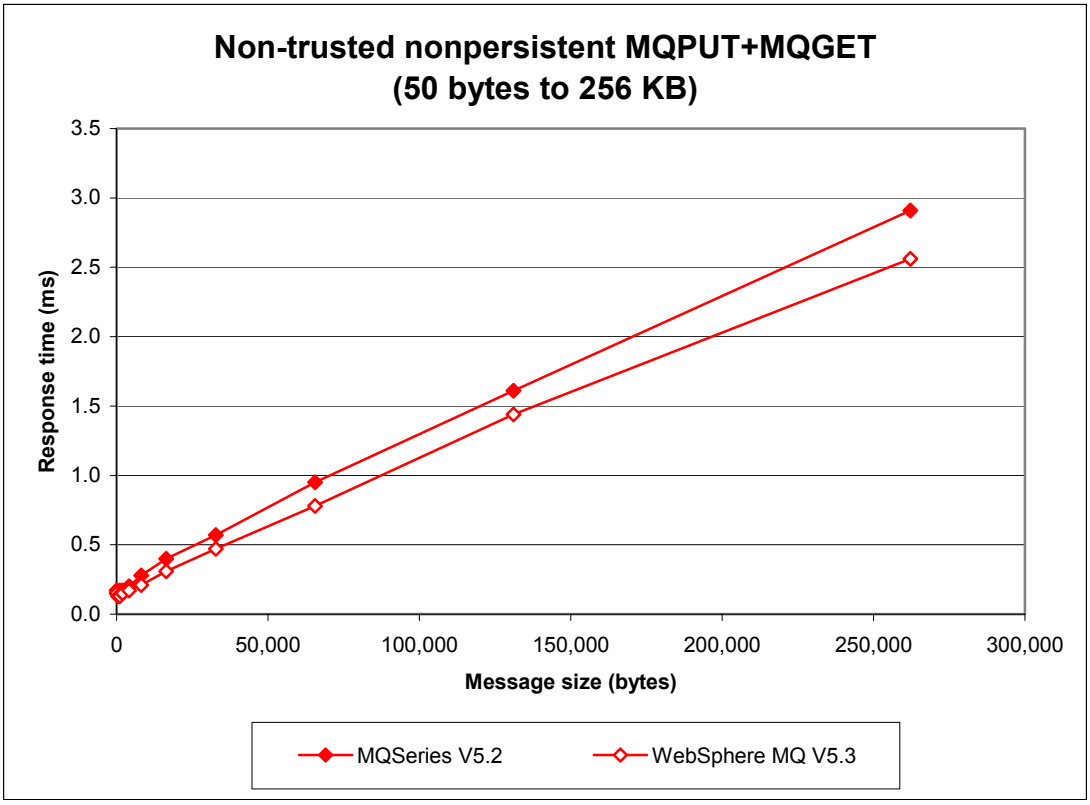

Figure 17 – The effect of nonpersistent message size on MQI response time (50 bytes to 256 KB)

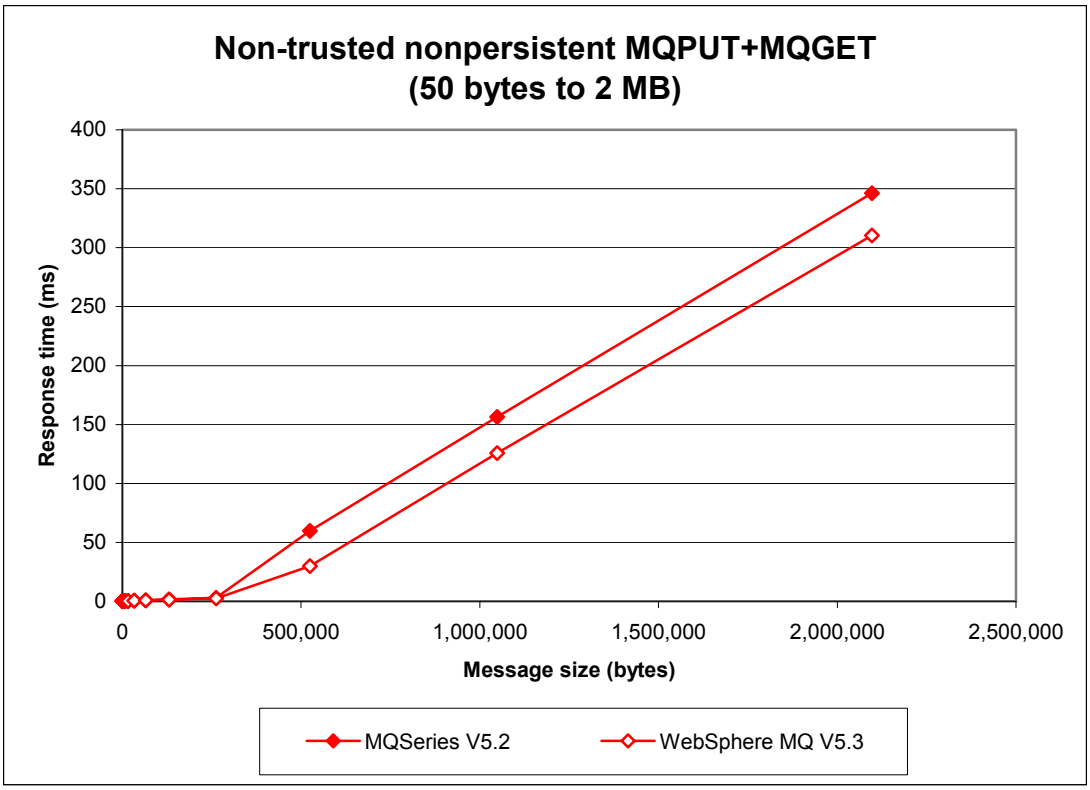

Figure 18 – The effect of nonpersistent message size on MQI response time (50 bytes to 2 MB)

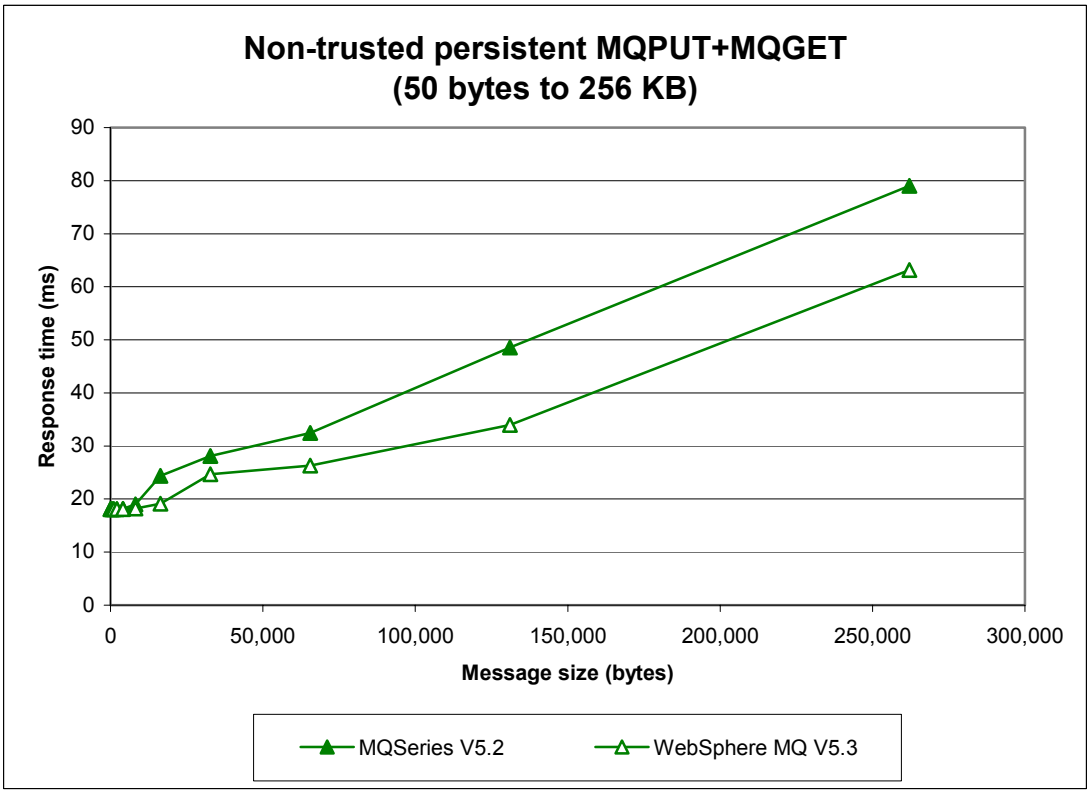

Figure 19 – The effect of persistent message size on MQI response time (50 bytes to 256 KB)

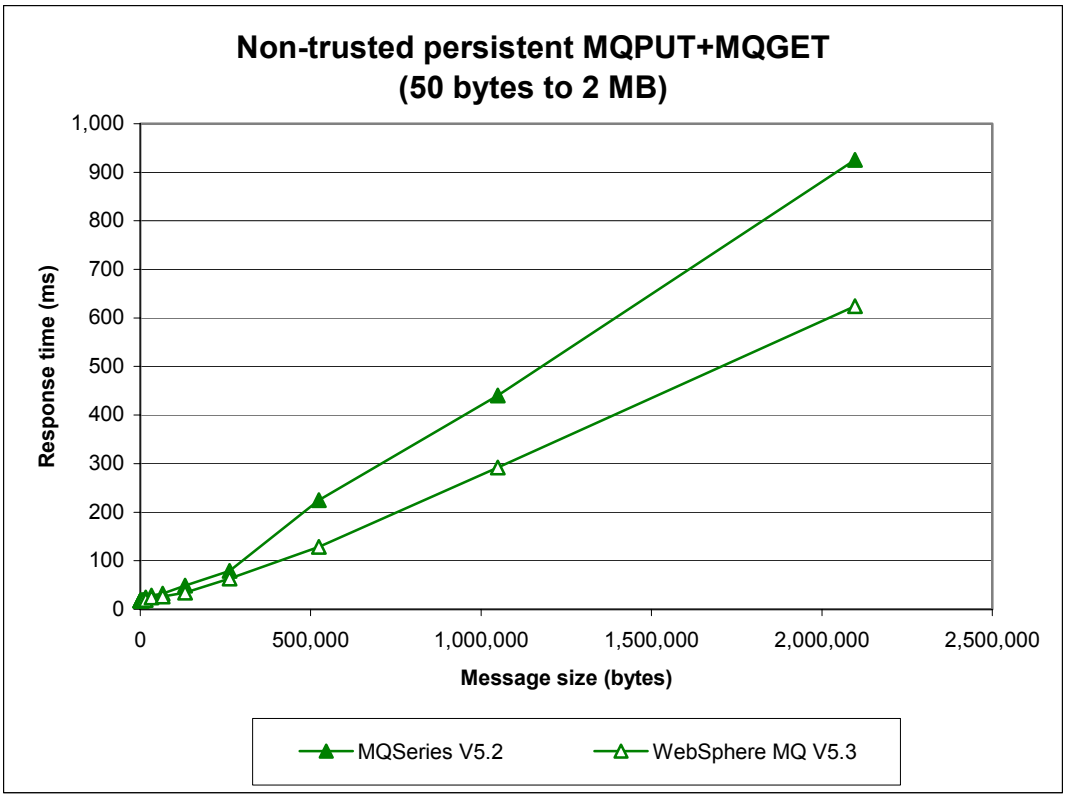

Figure 20 – The effect of persistent message size on MQI response time (50 bytes to 2 MB)

In the local queue manager, client channel and distributed queuing scenarios that follow, 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=4095, LogBufferPages=512`. (Note that WebSphere MQ V5.3 can have a maximum value for *LogFileSize* of 16384, whereas the maximum value for MQSeries V5.2 is only 4095 on Windows NT and Windows 2000. The smaller value was chosen for the 200 KB tests to permit direct comparison between WebSphere MQ V5.3 and MQSeries V5.2.)

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

Test name	Apps	Message size (KB)	Round trips/s	% change over MQSeries V5.2	Response time (s)
local_np1	4 (3)	2	6,393 (4,991)	+28	0.001 (0.001)
local_np1_20K	4 (3)	20	2,832 (2,251)	+26	0.001 (0.001)
local_np1_200K	2 (1)	200	241 (197)	+22	0.009 (0.005)
local_pm3	72 (108)	2	1,177 (322)	+266	0.072 (0.419)
local_pm3_20K	32 (44)	20	288 (129)	+123	0.133 (0.428)
local_pm3_200K	28 (12)	200	32 (16)	+100	0.984 (0.858)

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

Note: The figures on the top row in each cell show the peak throughput obtained for WebSphere MQ V5.3. The figures in parentheses show the corresponding values for MQSeries V5.2.

Figure 21 below shows that the throughput of small nonpersistent messages has improved significantly through a local queue manager in WebSphere MQ V5.3.

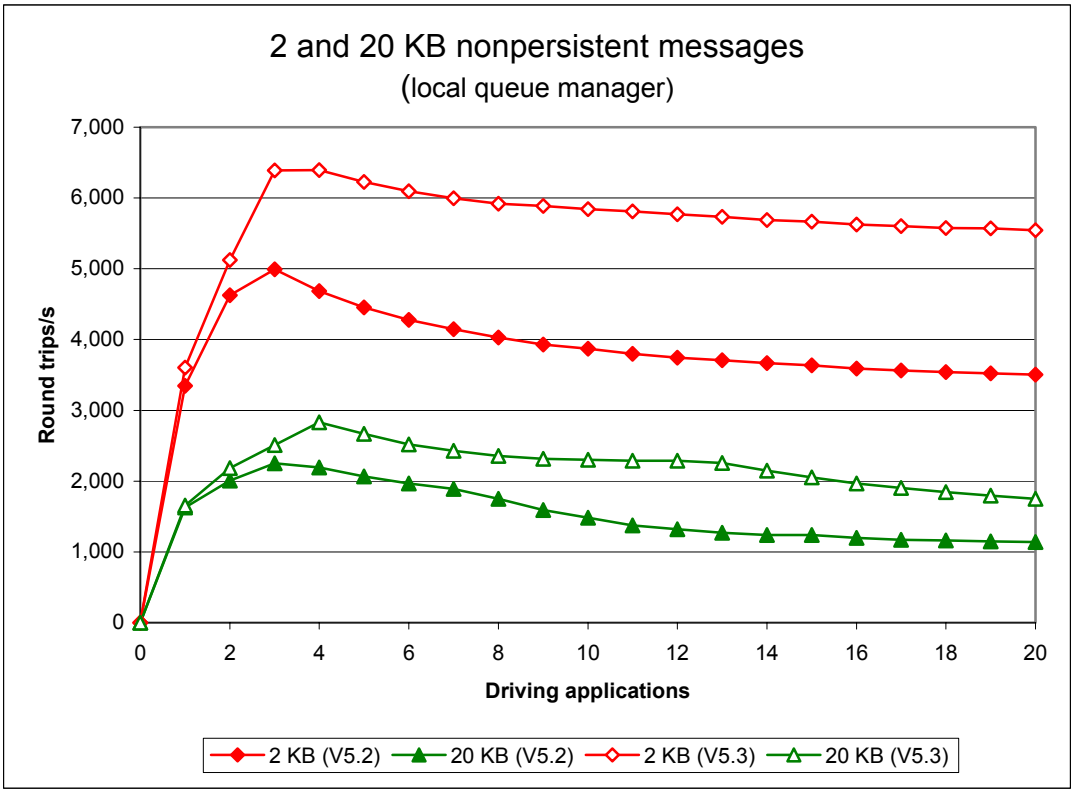

Figure 21 – 2 and 20 KB nonpersistent messages, local queue manager

Figure 22 below shows that the throughput of 20 KB, and especially 2 KB, persistent messages have improved significantly through a local queue manager in WebSphere MQ V5.3.

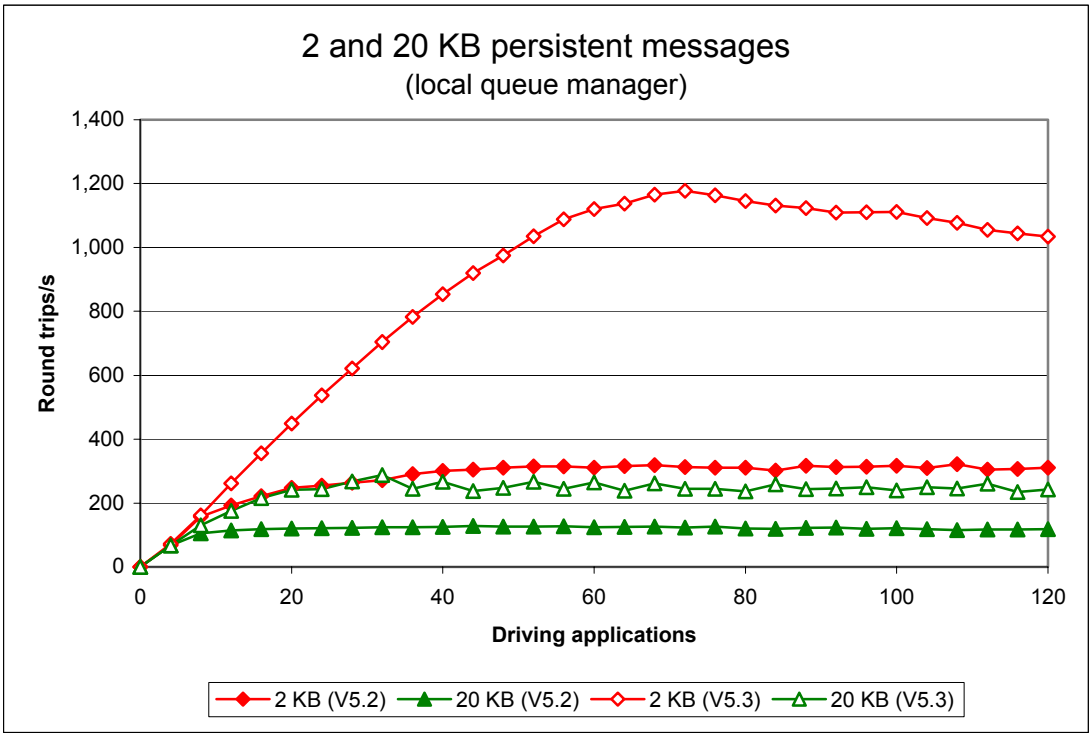

Figure 22 – 2 and 20 KB persistent messages, local queue manager

Figure 23 below shows that the throughput of 200 KB nonpersistent and persistent messages through a local queue manager have improved in WebSphere MQ V5.3. Note: Response time had exceeded one second at 16 or more and 32 or more driving applications for MQSeries V5.2 and WebSphere MQ V5.3 respectively in the persistent message tests. The tests were terminated when several successive measurements yielded response times exceeding one second.

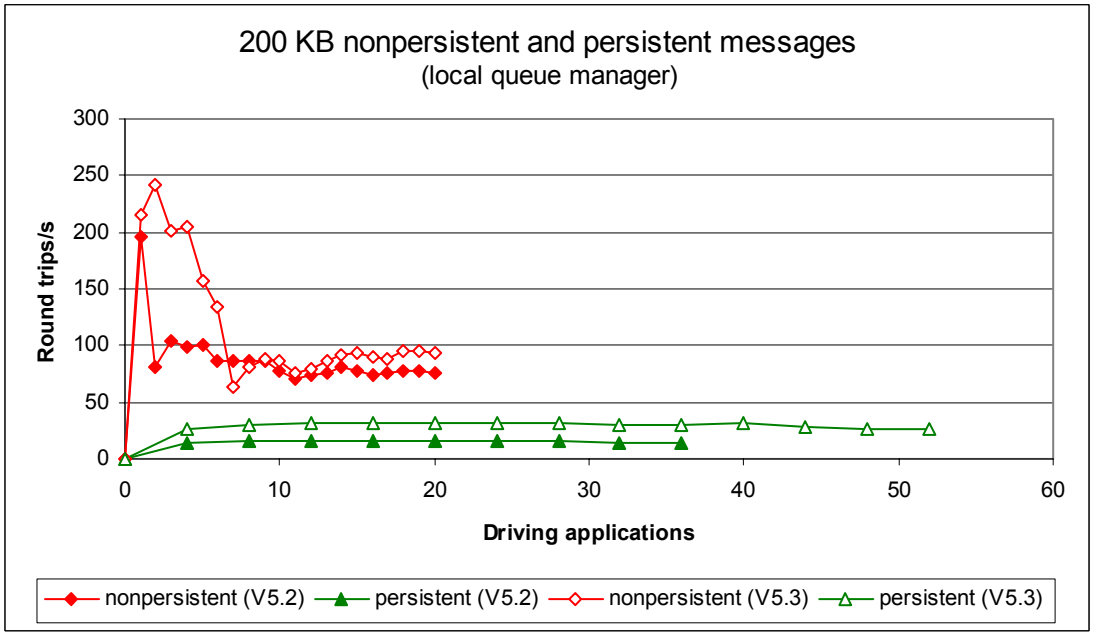

Figure 23 – 200 KB nonpersistent and persistent messages, local queue manager

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

Test name	Apps	Message size (KB)	Round trips/s	% change over MQSeries V5.2	Response time (s)
clnp1	9 (7)	2	3,925 (3,162)	+24	0.003 (0.003)
clnp1_20K	17 (10)	20	1,211 (984)	+23	0.017 (0.012)
clnp1_200K	9 (20)	200	67 (27)	+148	0.156 (0.901)
clpm3	64 (100)	2	1,049 (338)	+210	0.074 (0.351)
clpm3_20K	60 (64)	20	281 (125)	+125	0.278 (0.558)
clpm3_200K	20 (28)	200	29 (15)	+93	0.771 (2.163)

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

Note: The figures on the top row in each cell show the peak throughput obtained for WebSphere MQ V5.3. The figures in parentheses show the corresponding values for MQSeries V5.2.

Figure 24 below shows that the throughput of small nonpersistent messages through client channels has improved significantly in WebSphere MQ V5.3.

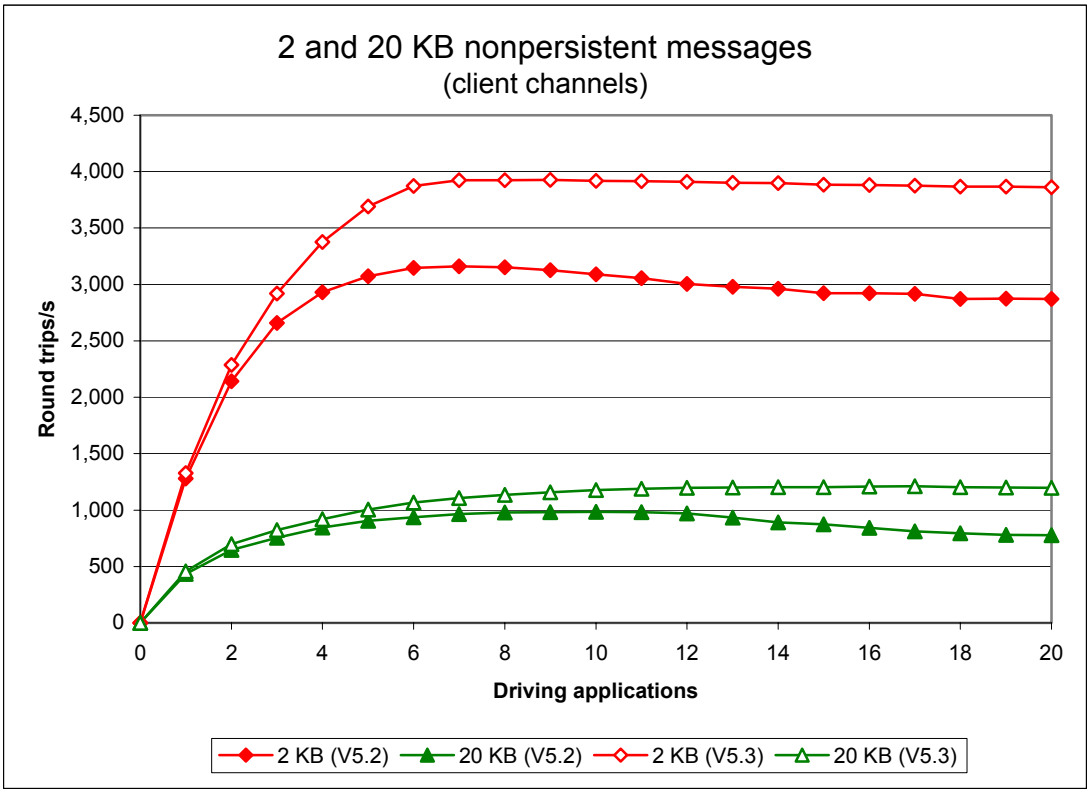

Figure 24 – 2 and 20 KB nonpersistent messages, client channels

Figure 25 below shows that the throughput of 2 and 20 KB persistent messages through client channels have improved significantly in WebSphere MQ V5.3.

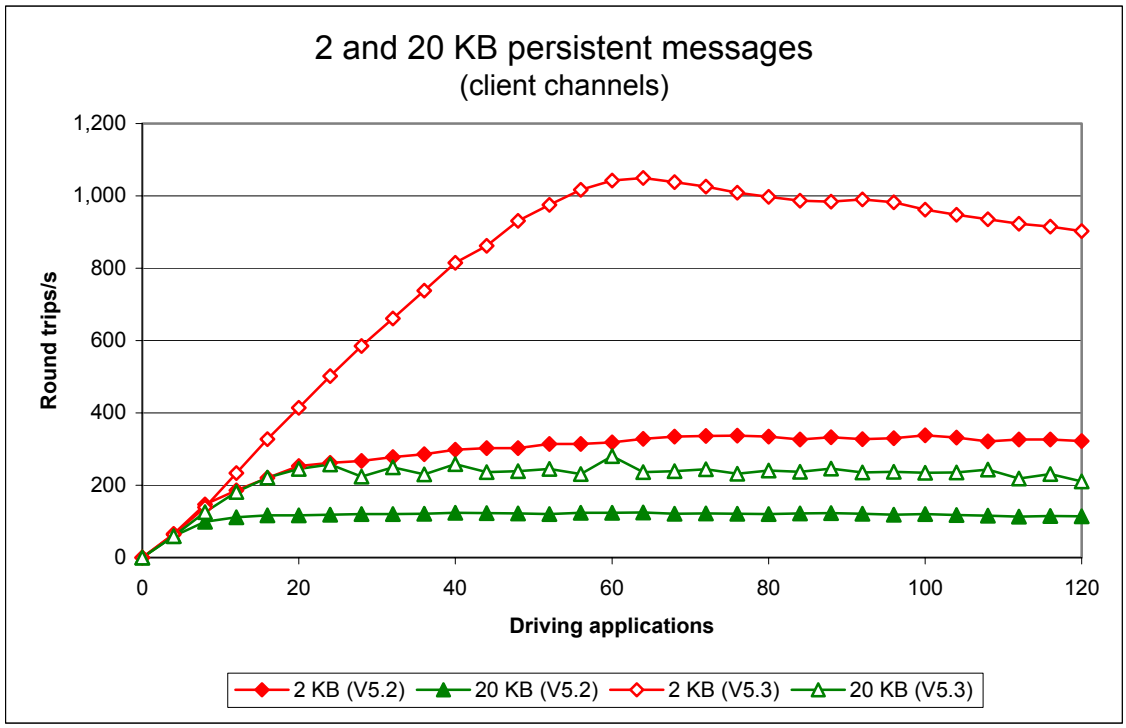

Figure 25 – 2 and 20 KB persistent messages, client channels

Figure 26 below shows that the throughput of 200 KB nonpersistent and persistent messages through client channels have improved in WebSphere MQ V5.3. Note: Response time had exceeded one second at 8 or more and 28 or more driving applications for MQSeries V5.2 and WebSphere MQ V5.3 respectively in the persistent message tests. The tests were terminated when several successive measurements yielded response times exceeding one second.

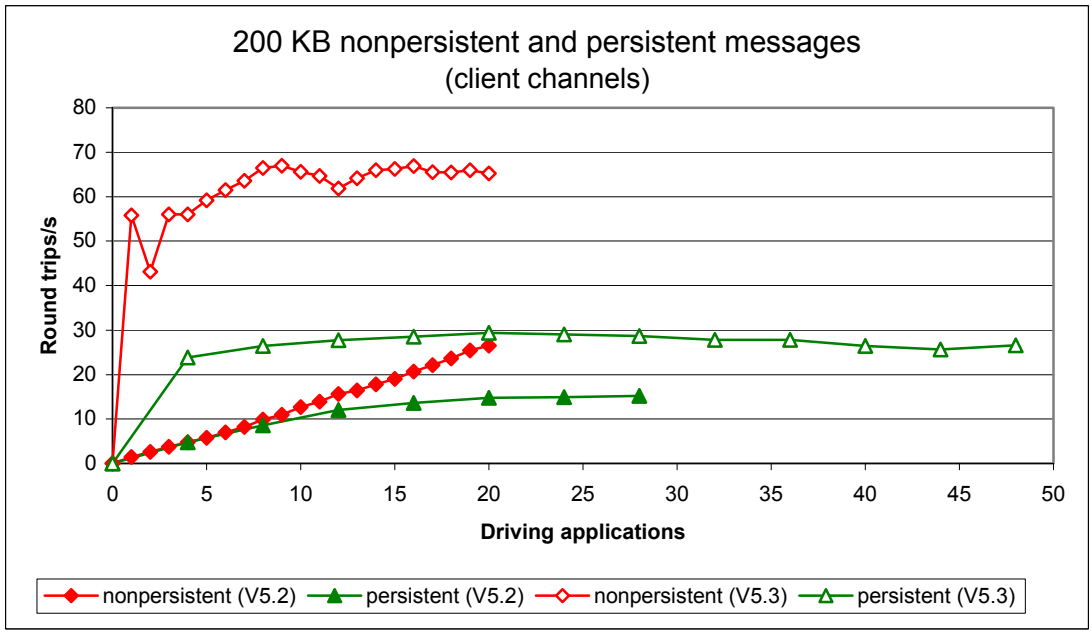

Figure 26 – 200 KB nonpersistent and persistent messages, client channels

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

Test name	Apps	Message size (KB)	Round trips/s	% change over MQSeries V5.2	Response time (s)
dqnp1	15 (20)	2	4,722 (3,740)	+26	0.003 (0.007)
dqnp1_20K	10 (14)	20	1,296 (1,223)	+6	0.009 (0.014)
dqnp1_200K	18 (15)	200	82 (83)	-1	0.255 (0.210)
dqpm1	120 (120)	2	783 (418)	+87	0.177 (0.346)
dqpm1_20K	108 (100)	20	265 (110)	+141	0.457 (1.053)
dqpm1_200K	12 (8)	200	21 (12)	+200	0.632 (0.761)

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

Note: The figures on the top row in each cell show the peak throughput obtained for WebSphere MQ V5.3. The figures in parentheses show the corresponding values for MQSeries V5.2.

Figure 27 below shows that the throughput of 2 KB nonpersistent messages through server channels has improved significantly in WebSphere MQ V5.3. Throughput of 20 KB messages was similar to that for MQSeries V5.2.

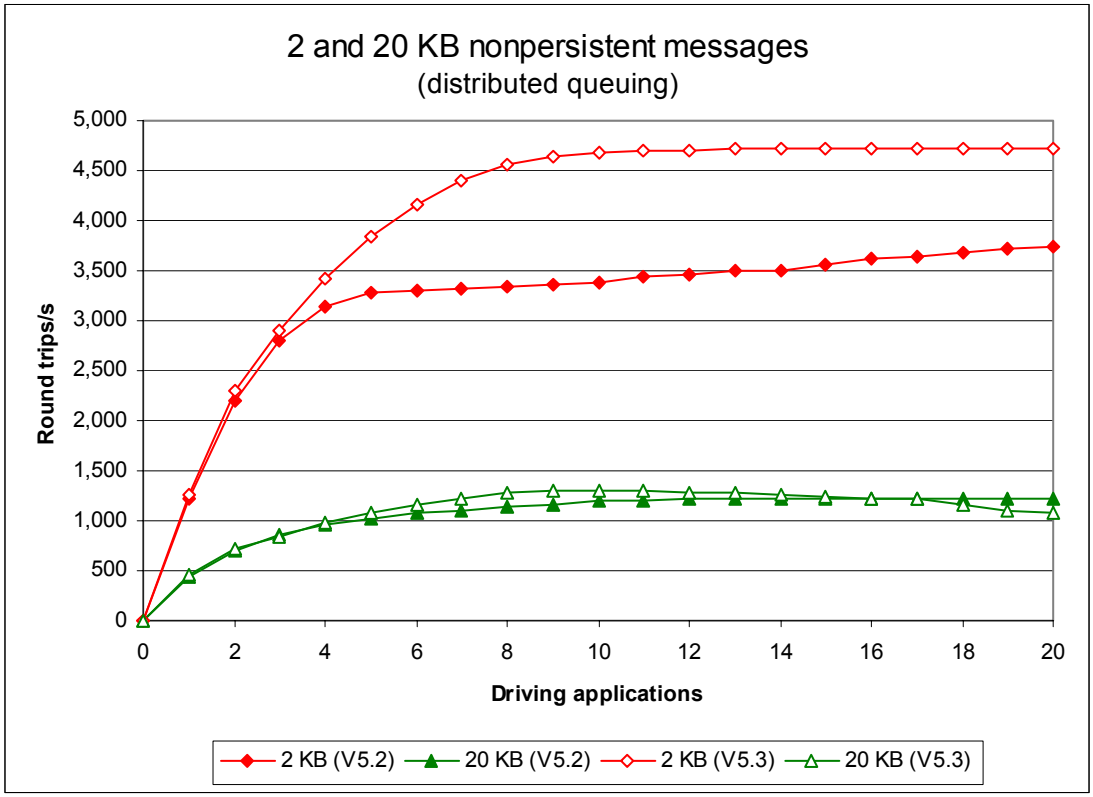

Figure 27 – 2 and 20 KB nonpersistent messages, server channels

Figure 28 below shows that the throughput of both 2 and 20 KB persistent messages through server channels have improved significantly in WebSphere MQ V5.3.

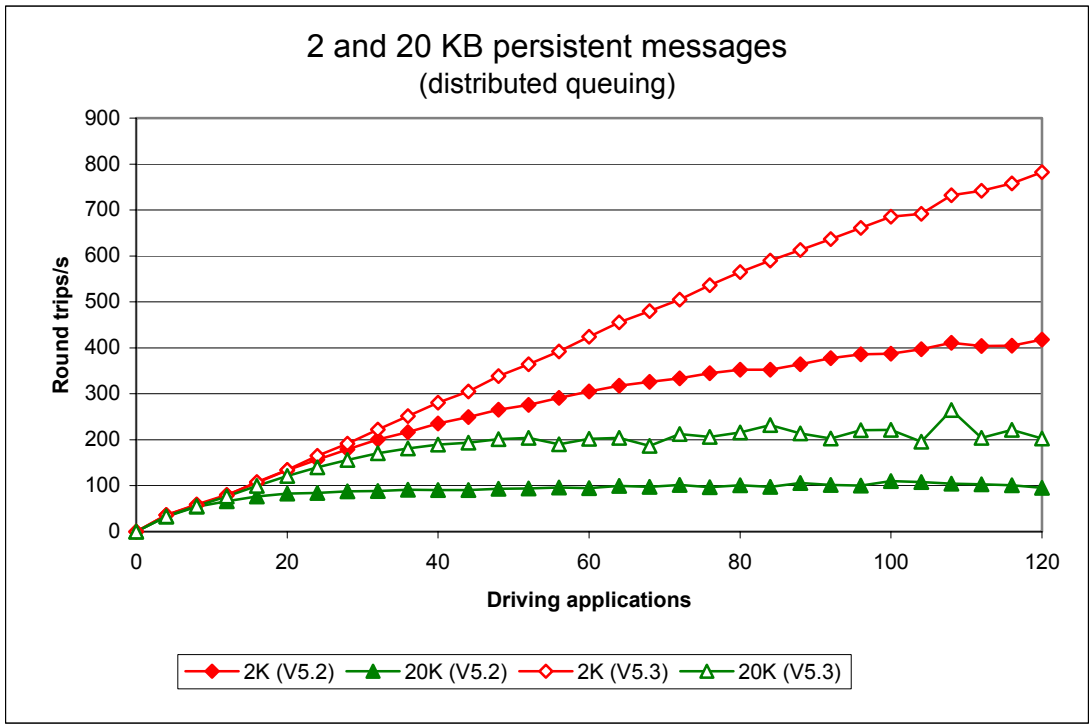

Figure 28 – 2 and 20 KB persistent messages, server channels

Figure 29 below shows that the throughput of 200 KB nonpersistent messages through server channels has improved slightly, and the throughput of 200 KB persistent messages has improved more significantly, in WebSphere MQ V5.3. Note: Response time had exceeded one second at 12 or more and 16 or more driving applications for MQSeries V5.2 and WebSphere MQ V5.3 respectively in the persistent message tests. The tests were terminated when several successive measurements yielded response times exceeding one second.

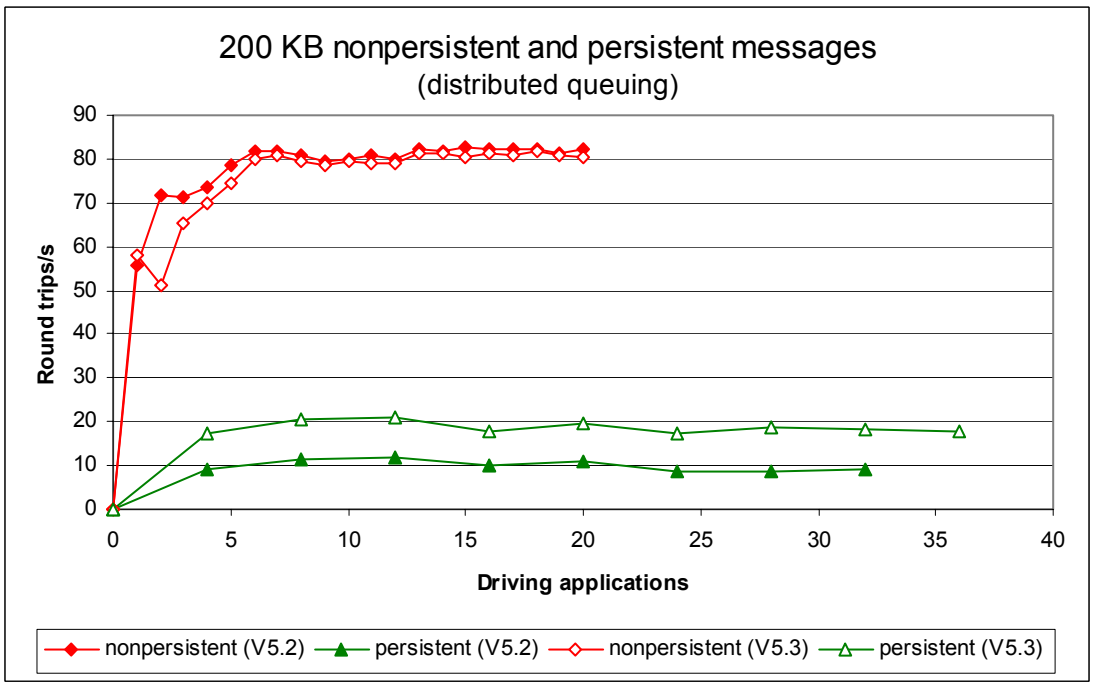

Figure 29 – 200 KB nonpersistent and persistent messages, server channels

4 Trusted server application

Test name	Scenario	Message type	Apps	Round trips/s	% change over MQSeries V5.2	Response time (s)
local_np1_trusted	Local	Nonpersistent	3 (2)	9,421 (8,117)	+16	< 0.001 (< 0.001)
local_pm3_trusted	Local	Persistent	76 (72)	1,248 (334)	+274	0.072 (0.257)
clnp1_trusted	Client	Nonpersistent	10 (8)	4,588 (4,058)	+13	0.002 (0.003)
clpm3_trusted	Client	Persistent	68 (84)	1,121 (352)	+218	0.068 (0.258)
dqnp1_trusted	Server	Nonpersistent	12 (7)	5,499 (4,286)	+28	0.002 (0.002)
dqpm1_trusted	Server	Persistent	120 (120)	787 (431)	+83	0.183 (0.345)

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

Note: the large figures in **Table 14** above are for WebSphere MQ V5.3, the smaller figures in parentheses are for MQSeries V5.2. Note that throughput was still increasing with the number of driving applications for both versions of the product when the 'dqpm1_trusted' test was completed at 120 applications.

Figure 30 below shows the improvement in throughput achieved for nonpersistent messages in the local queue manager scenario when the server application was run trusted ('fastpath').

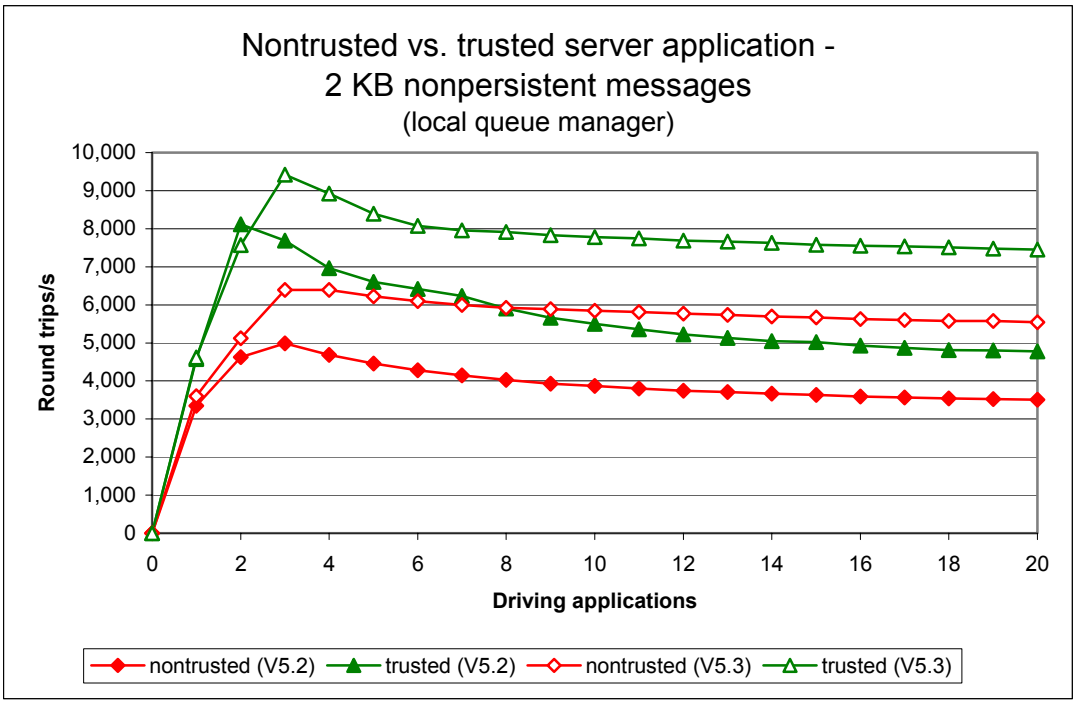

Figure 30 – Non-trusted vs trusted server application, local queue manager, 2KB nonpersistent messages

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,
- 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 31** below with **Figure 10**.

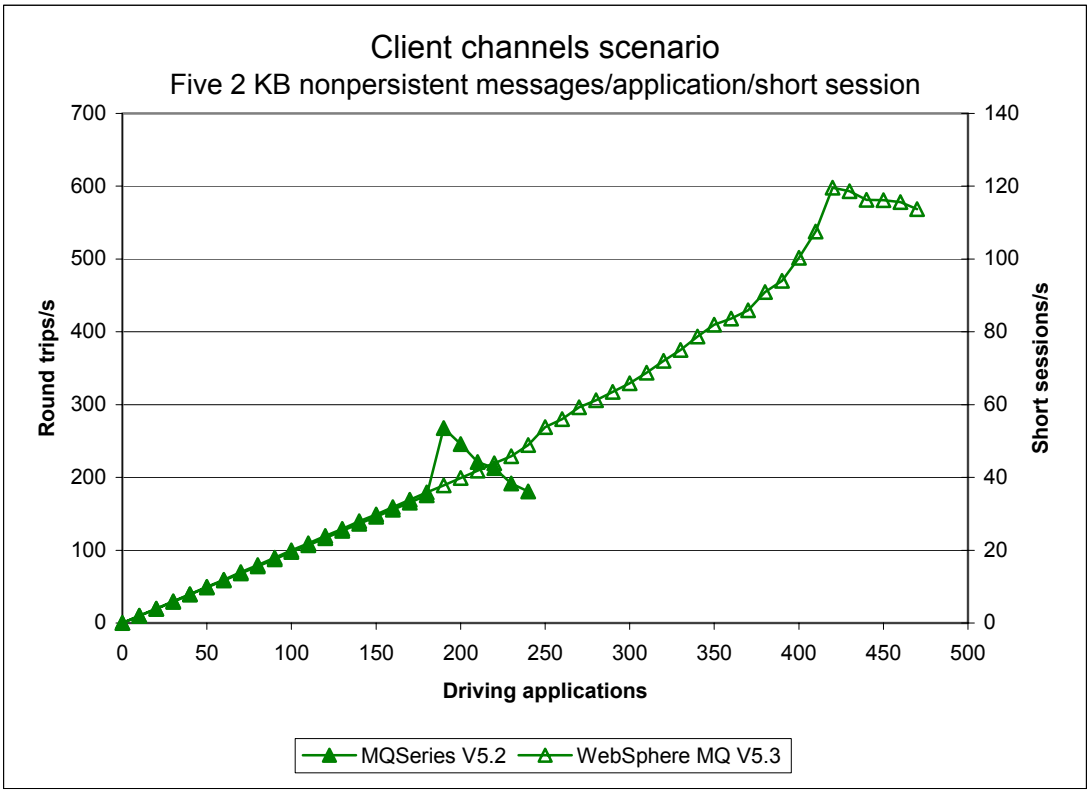

Figure 31 – Short sessions, client channels

Test name	Apps	Round trips/s	Short sessions/s	Response time (s)
clnp1_ss_r3600_runmqlsr	410 (180)	538 (176)	108 (35)	0.603 (0.018)

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

Note: the large figures in **Table 15** above are for WebSphere MQ V5.3, the smaller figures in parentheses are for MQSeries V5.2. 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.

Test name	Common reply queue?	Max-Channels	Apps	Rate /app/h	Round trips/s	Response time (s)
clnp1	Yes	5,000	9	n/a	3,925	0.003
clnp1_r3600_runmqlsr	Yes	5,000	1,350	3,600	1,349	0.010
clnp1_c6000_runmqlsr_t10	Yes	8,192	6,000	150	267	0.645
clnp1_cmax_runmqlsr_t10_ no_correlid	No	50,000	10,800 (11,800)	60	179 (141)	0.742 (26.1)
clnp1_cmax_runmqlsr_t10	Yes	50,000	11,500 (13,000)	60	191 (140)	0.372 (38.1)

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

Notes:

i. 'MaxChannels' refers to the Windows registry key under the 'Channels' section of the registry entry for the queue manager.

ii. '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.

iii. Higher throughputs were observed in the 'clnp1_c6000_runmqlsr_t10' test but the corresponding response times were in excess of one second.

iv. All the above tests used a common reply queue except for 'clnp1_cmax_runmqlsr_t10_no_correlid', which used one reply queue for each client.

v. Two Windows 2000 machines were used for the first three tests but the driver machine had to be replaced with a more powerful AIX M80 machine for the last two ('cmax') tests; see **Measurement environment** for more details.

Table 16 above shows the number of clients at which maximum throughput was achieved under different conditions. All client connections were made trusted. **Figure 32** below shows how the maximum throughput that could be achieved decreased with the number of clients connected. Two rows of data are shown in **Table 16** for the last two ('cmax') tests. The first row corresponds to the number of clients at which maximum throughput was observed. These are the data plotted in **Figure 32**. The second row shows the largest number of clients that could be successfully connected regardless of throughput and response time. Note that the server machine still had considerable available memory at the end of the two 'cmax' tests. Thus only 2,985 MB had been committed at 11,800 driving applications for the 'clnp1_cmax_runmqlsr_t10_no_correlid' test; and 2,249 MB had been committed at 13,000 driving applications for the 'clnp1_cmax_runmqlsr_t10' test.

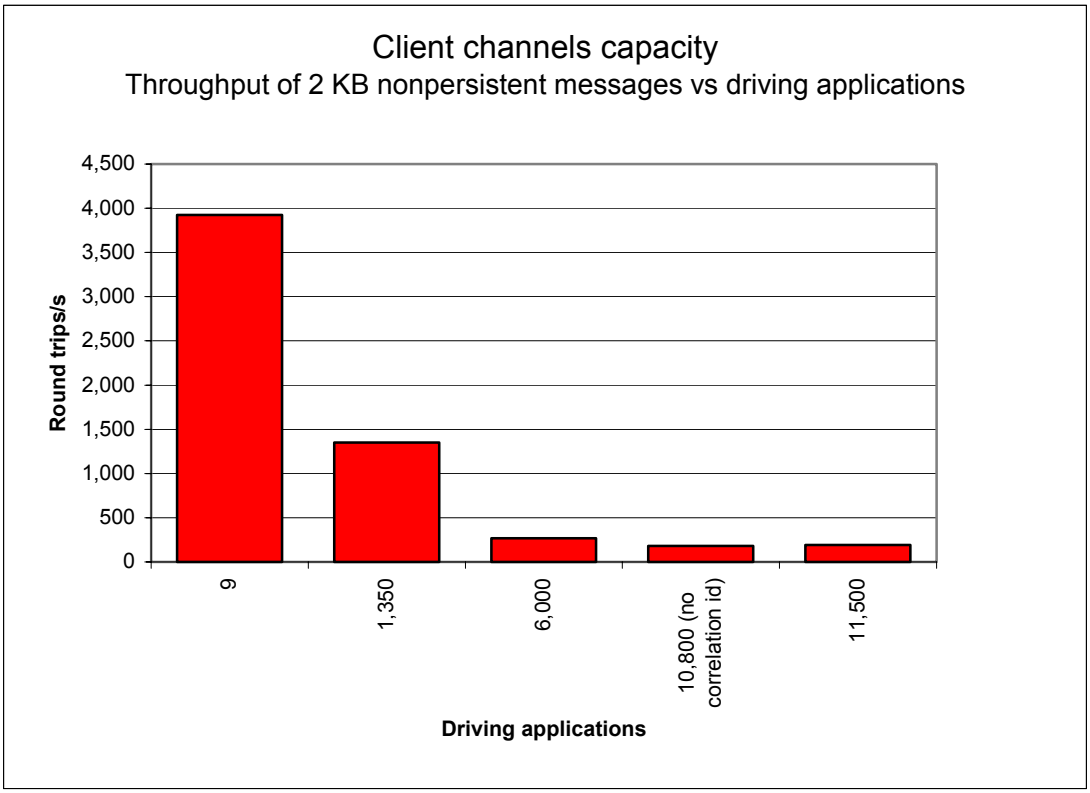

Figure 32 – Effect of number of client channels on message throughput

By comparing the amount of memory ('committed bytes') at 11,000 client connections in the two 'cmax' tests and then dividing the difference by 11,000, a storage cost per open queue of approximately 36 KB was obtained for WebSphere MQ V5.3. This compares with a much larger storage cost per open queue of approximately 340 KB in MQSeries V5.2 (figure taken from SupportPac MP76: *MQSeries for Windows NT and Windows 2000 V5.2 – Performance Highlights* V1.2). Further calculations showed that a trusted client connection required approximately 127 KB of memory in addition to the storage requirements of a queue manager and any queues.

6.2 Distributed queuing– capacity measurements

The measurements in this section illustrate the trade-off of workload (message rate) through server channel pairs connecting two queue managers against the total number of server channel pairs.

For the first three tests in **Table 17** below the queue manager logs and the *MaxChannels* registry key (in the *Channels* section of the queue managers' registry entries) were configured as follows:

```
LogPrimaryFiles=3, LogFileSize=64, LogBufferPages=17, MaxChannels = 5000
```

However when this configuration was used in a pilot run of the 'dqnp1_qmax_runmqlsr' test, WebSphere MQ reported that the log space was too small to hold the 12,000 channel definitions required by each of the queue managers (6,000 each for the sender and receiver channels). More, and larger, logs were therefore used for the final run of this test:

```
LogPrimaryFiles=12, LogFileSize=16384, LogBufferPages=512, MaxChannels =
20000
```

Although all the tests described in this report used the default method of circular logging, we would have expected to have needed larger logs to hold the channel definitions had we used linear logging instead. Note, however, that in a clustered environment (not tested here) the

majority of channels are auto-defined and therefore less overall log space would be required to store the channel definitions.

Test name	Server channel pairs	Apps	Rate /app/h	Round trips/s	Response time (s)
dqnp1	4	15	n/a	4,722	0.003
dqnp1_q1000_runmqlsr	1,000	1,000	5,250	1,454	0.004
dqnp1_r3600_runmqlsr	4	1,550	3,600	1,546	0.027
dqnp1_qmax_runmqlsr	6,000	6,000	60	100	0.225

Table 17 – Capacity measurements, 2 KB nonpersistent messages, server channels

Note: Two Windows 2000 machines were used for the first three tests but the driver machine had to be replaced with a more powerful AIX S80 machine for the last ('qmax') test; see **Measurement environment** for more details.

Table 17 above shows the number of driving applications at which maximum throughput was achieved under different conditions. **Figure 33** below corresponds to this table and shows how the maximum throughput that could be achieved decreased with the number of driving applications connected. Note that the final test, 'dqnp1_qmax_runmqlsr', was arbitrarily stopped at 6,000 applications with no sign of the server box constraining due to slow response times.

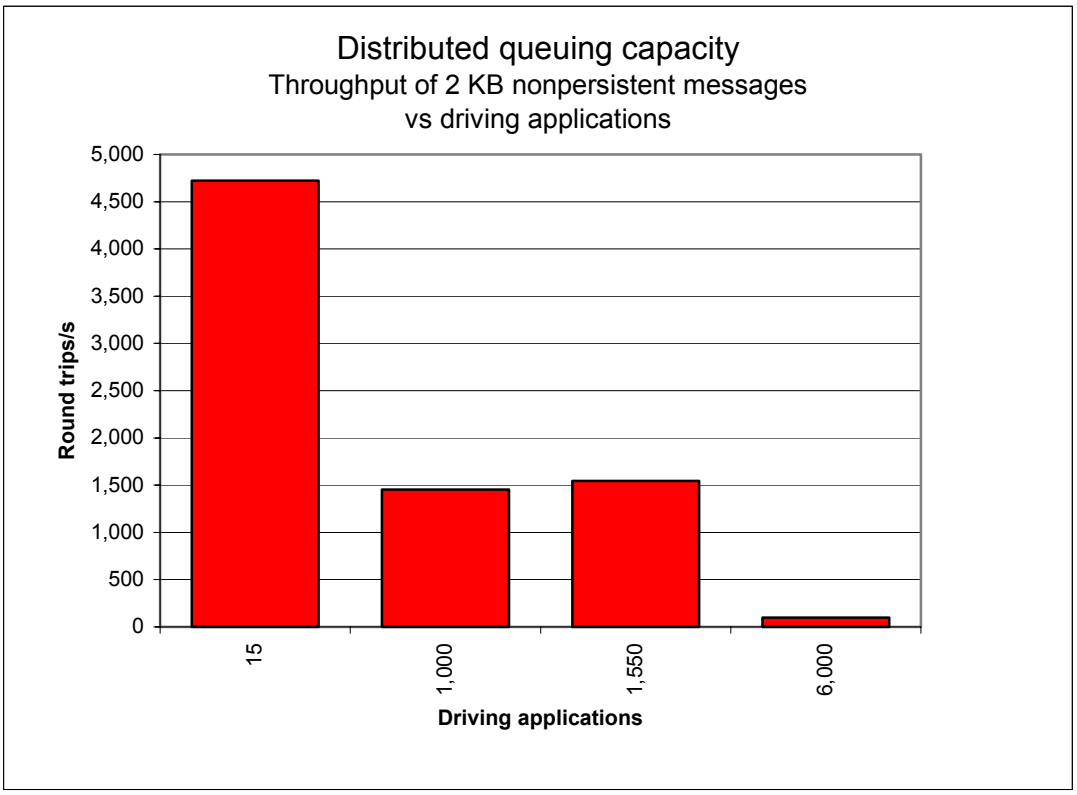

Figure 33 – Effect of number of driving applications on message throughput, server channels

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.

7.3 Tuning the operating system (Windows 2000)

7.3.1 Number of ephemeral TCP ports: 'MaxUserPort'

The default maximum number of ephemeral TCP ports on Windows 2000 is 5000, the first 1024 of which are normally reserved for system use. Therefore a Windows 2000 machine cannot normally be used to drive tests requiring more than about 4000 clients. However, the Windows registry entry, *MaxUserPort*, may be used to specify the maximum port number to use when an application requests any available user port from the system. The valid range of values is 5000 to 65534, and the default is 5000. Increasing the value of *MaxUserPort* up to 65534 therefore allows more clients to be connected, assuming other resources are not limiting. *MaxUserPort* resides in the registry key:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`; see the Microsoft Knowledge Base article Q196271 for further information.

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
Memory (RAM)	8 GB
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

An IBM M80 was used as the driving applications machine for the two maximum client channels tests, ‘clnp1_cmax_runmqlsr_t10’ and ‘clnp1_cmax_runmqlsr_t10_no_correlid’ referred to in **Table 16** (one of the Windows machines described above was used as the server machine):

Machine model	7026-M80
Processor	500 Mhz RS64 III
Architecture	4-way SMP
Memory (RAM)	4 GB
Disk	3 internal SCSI (9GB each, 1 O/S, 1 O/S + swap)
Network	1 Gigabit Ethernet

Note that AIX uses CCSID 819 whereas Windows uses CCSID 437 and therefore some CPU resources would have been consumed performing data conversion between the two operating systems.

An IBM S80 was used as the driving applications machine for the maximum server channel pairs test, ‘dqnp1_qmax_runmqlsr’ referred to in **Table 17** (one of the Windows machines described above was used as the server machine):

Machine model	7017-S80
Processor	375 MHz PowerPC RS64-III
Architecture	24-way SMP IBM SSA 160 SerialRAID Adapter
Memory (RAM)	32 GB
Disk	2 internal 16 bit 9.1 GB LVD SCSI disks (1 O/S, 1 O/S + swap)
	One 9.1 GB physical SSA160 disk divided into 3 logical disks: 1 swap, 1 queue, 1 log. (Note: /usr/samples/kernel/vmtune –c 0)
Network	1 Gigabit Ethernet

Note that AIX uses CCSID 819 whereas Windows uses CCSID 437 and therefore some CPU resources would have been consumed performing data conversion between the two operating systems.

8.2 Software

For the IBM Netfinity 8500R machines:

Operating System	Microsoft Windows 2000 Advanced Server with Service Pack 2
MQSeries/WebSphere MQ	MQSeries for Windows NT and Windows 2000, Version 5.2 GA (plus efix 54142, subsequently available in CSD 1) WebSphere MQ for Windows, Version 5.3
Compiler	Microsoft Visual C++ 6.0 Professional Edition

Note: *MaxUserPort* = 35,000 for both the driver and server machines (see **Number of ephemeral TCP ports: 'MaxUserPort'**, page 34, for further details).

For the IBM M80:

Operating System	AIX 4.3.3
MQSeries/WebSphere MQ	Version 5.3, and Version 5.2 (Note: queue manager CCSID = 819)
Compiler	C for AIX Compiler, Version 5 (5.0.1.0)

For the IBM S80:

Operating System	AIX 4.3.3.0
MQSeries/WebSphere MQ	Version 5.3, and Version 5.2 (Note: queue manager CCSID = 819)
Compiler	C for AIX Compiler, Version 5 (5.0.1.0)

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.
```

A *LogFileSize* of 4095 rather than 16384 pages was chosen to allow direct comparison of WebSphere MQ V5.3 with MQSeries V5.2 (4095 pages is the maximum permissible value for MQSeries V5.2 on Windows NT and Windows 2000). All the tests described in this report used circular logging.

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).

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.

9 Glossary

Apps	The number of driving applications connected to the queue manager at the point corresponding to the reported performance measurement.
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.
Round trips/s	The average achieved message throughput rate (request messages per second) of all the driving applications together, measured by the server application.
% change over MQSeries V5.2	The percentage improvement in round trips/s for WebSphere MQ V5.3 compared to MQSeries V5.2.
Response time (s)	The average duration in seconds for each round trip, as measured and averaged by all the driving applications.
Zero think-time	The driving application sent a new request message as soon as it had received the reply to its previous request message.