WebSphere MQ for AIX V53 - Performance Evaluations

Version 1.1

5 August 2002

Richard Eures. B.Sc, M.Ed. IBM Certified

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”.

First Edition, July 2002
Second Edition, August 2002

This edition applies to V1.0 of WebSphere MQ for AIX V5.3 – Performance Evaluations and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

This report is intended to help the reader understand the performance characteristics of WebSphere MQ for AIX V5.3. 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 reader to evaluate the information and project the results to their operational environment.

The performance measurements included in this report were 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

UNIX is a registered trademark and licensed exclusively through X/Open Company Limited.

Preface

Target audience

This SupportPac is designed for people who:

- Will be designing and implementing solutions using WebSphere MQ for AIX.
- Want to understand the performance limits of WebSphere MQ for AIX V5.3.
- Want to understand what actions may be taken to tune WebSphere MQ for AIX.

The reader should have a general awareness of the AIX operating system and of MQSeries in order to make best use of this SupportPac. Readers should read the section '**How this document is arranged**'—**Page V** to familiarise themselves with where specific information can be found for later reference.

The contents of this SupportPac

This SupportPac includes:

- Release highlights performance charts.
- Performance measurements with figures and tables to present the performance capabilities of WebSphere MQ local queue manager, client channel, and distributed queuing scenarios.
- Interpretation of the results and implications on designing or sizing of the WebSphere MQ local queue manager, client channel, and distributed queuing 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 more useful manner? 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 to your local IBM Representative or Support Centre.

Acknowledgements

- The author is very grateful to Alexander Russell for help in producing this report.

Introduction

The three scenarios used in this report to generate the performance data are:

- Local queue manager scenario.
- Client channel scenario.
- Distributed queuing scenario.

The performance improvements in WebSphere MQ V5.3 can be divided into two areas:

- Queue manager enhancements.
- Channel capacity enhancements.

The enhancements to the queue manager are apparent through many of the measurements in this report when comparing WebSphere MQ V5.3 against Version 5.2. Channel capacity enhancements are covered briefly in the "*Release Highlights"* section and in more detail towards the end of the report.

Unless otherwise specified, the standard message sized used for all the measurements in this report is 2K (2,048 bytes), trusted channels use the 'amqcrsta' (inetd) listener, and nontrusted runmqlsr server application are used.

An AIX (model 7026 – M80) 4-way 500MHz RS64 III with 4GB of RAM was used as the device under test for all the measurements.

How this document is arranged

Release highlights

Pages: 1-4

Section one outlines the major performance *improvements* achieved in WebSphere MQ V5.3 compared to Version 5.2. The highlights are a subset of the results shown in the *Performance headlines* section.

Performance headlines

Pages: 5-16

Section two contains the performance *headlines* for each of the three scenarios, with MQI applications connected to:

- A local queue manager.
- A remote queue manager over MQI-client channels.
- 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 connected to a queue manager.
- The maximum number of server channel pairs between two queue managers, for a fixed think time between messages until the response time exceeds one second.

Large messages

Pages: 17-24

Section three contains performance measurements for *large messages*. This includes *MQI response times* of 50byte to 2MB messages, and *20K* and *200K* messages using the same scenarios as for the "Performance Headlines".

Trusted server application

Page: 25

Section four contains performance measurements for a *trusted server* application, using the same three scenarios as for the "Performance headlines".

Short sessions

Page: 26- 27

Section five contains performance measurements for *short sessions*. That is, an MQI application connecting to the queue manager, processing a few messages between connecting to and disconnecting from the queue manager.

Capacity measurements

Pages: 28-31

Section six of this document shows:

- The number of MQI-client channels that were connected into a single queue manager, with a server application processing one nonpersistent round trip per MQI-client per minute.
- The number of server channel pairs that were connected between two queue managers on separate server machines, with a server application processing one nonpersistent round trip per server channel pair per minute.

Tuning recommendations

Pages: 32-35

In previous SupportPacs, tuning recommendations have been in a separate section. In this document, queue manager parameters are mentioned with the measurements they are appropriate to.

Measurement environment

Pages: 36-37

A summary of the way in which the workload is used in each test scenario is given in the "Performance headlines" section. This includes a more detailed description of the workload, hardware and software specifications.

Glossary

Page: 38

A short glossary of the terms used in the tables throughout this document.

CONTENTS

TABLES

FIGURES

1 Release highlights

1.1 Improvements to nonpersistent and persistent messaging

- Maximum nonpersistent message throughput increased by 6% in a local queue manager environment, 11% in an MQI-client environment, and 11% in a distributed queuing environment.
- Maximum persistent message throughput increased by 78% in a local queue manager environment, 66% in an MQI-client environment, and 64% in a distributed queuing environment.

1.2 Peak message throughput – local queue manager

Figure 1 shows the peak round trips per second achieved for nonpersistent and persistent messages with a local queue manager, MQSeries V5.2 compared to WebSphere MQ V5.3.

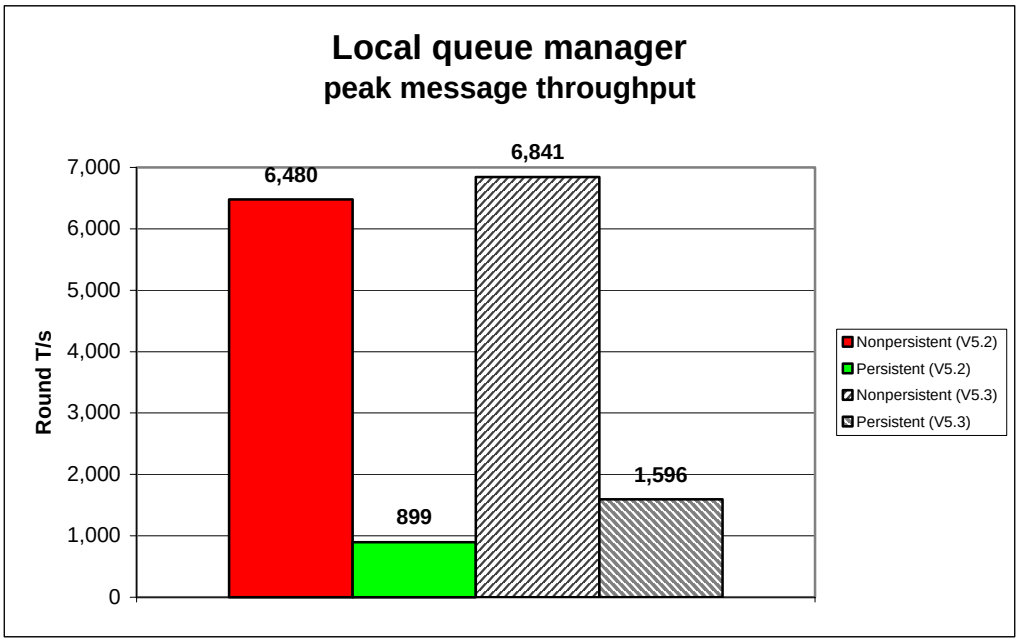

Figure 1 – Peak message throughput, local queue manager

Note: Messaging in these tests is with no think-time.

1.3 Peak message throughput – client channels

Figure 2 shows the peak round trips per second achieved for nonpersistent and persistent messages with MQI-client channels, MQSeries V5.2 compared to WebSphere MQ V5.3.

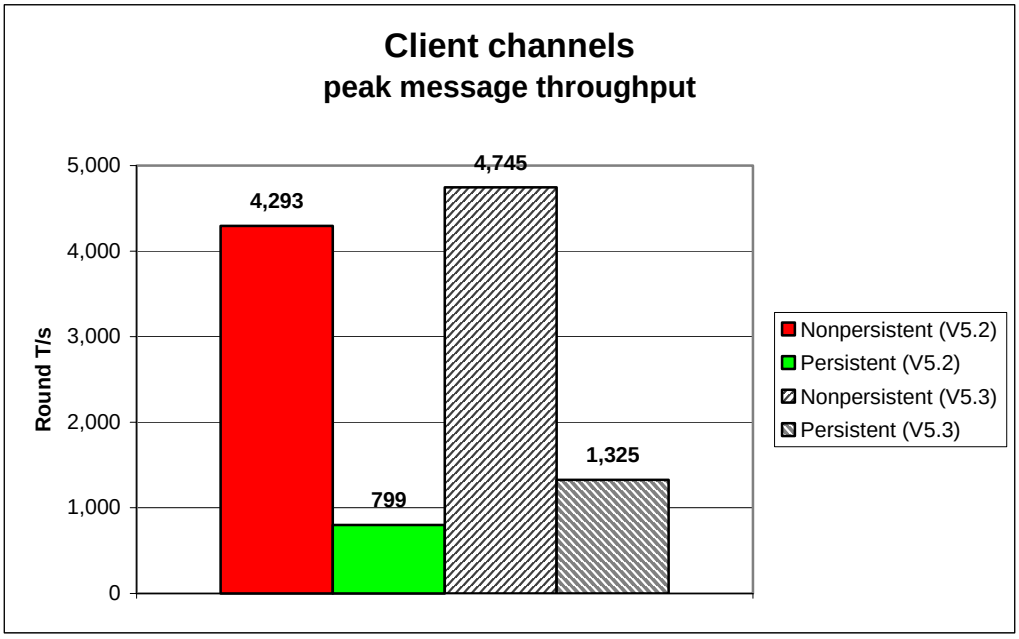

Figure 2 – Peak message throughput, client channels

Note: Messaging in these tests is with no think-time.

1.4 Peak message throughput – distributed queuing

Figure 3 shows the peak round trips per second achieved for nonpersistent and persistent messages with server channels, MQSeries V5.2 compared to WebSphere MQ for V5.3.

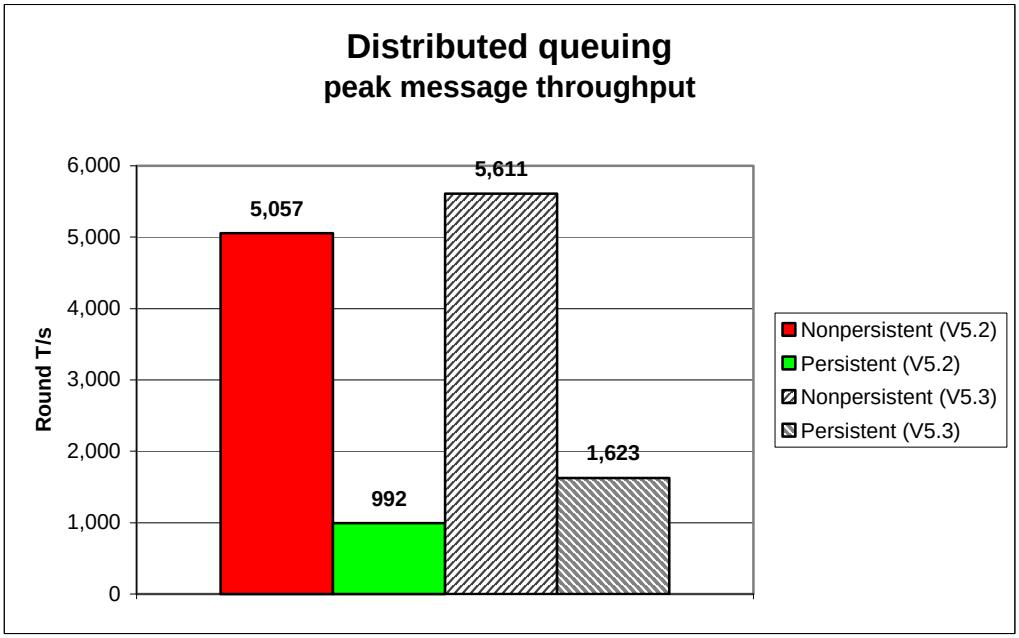

Figure 3 – Peak message throughput, distributed queuing

Note: Messaging in these tests is with no think-time.

1.5 Improvements to channel capacity limits

- Client channel (trusted MQI-client connections) increased from 6,500 to 34,500 using one reply queue for all clients.
- Distributed queuing (trusted server channels) increased from 1,600 to 6,900 pairs

1.6 Capacity limits – client channels

Figure 4 shows the maximum number of MQI-client connections made concurrently into a single queue manager on the server machine.

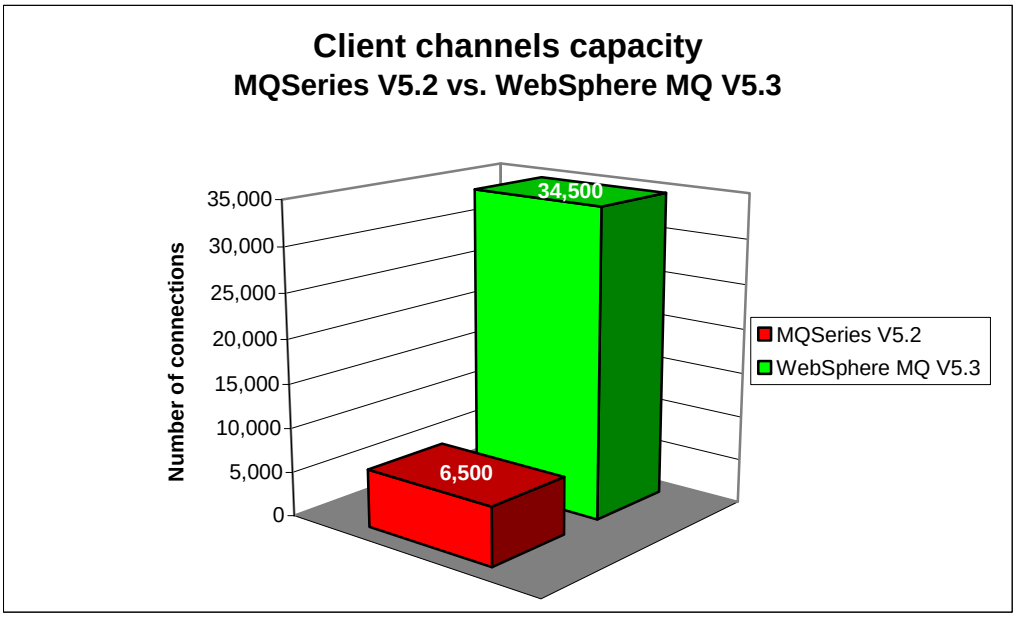

Figure 4 – Maximum number of client connections

Note: Messaging in these tests uses a rate of 1round trip per MQI-client per minute.

1.7 Capacity limits – distributed queuing (server channels)

Figure 5 below shows the maximum number of server channel pairs achieved between two queue managers on separate server machines.

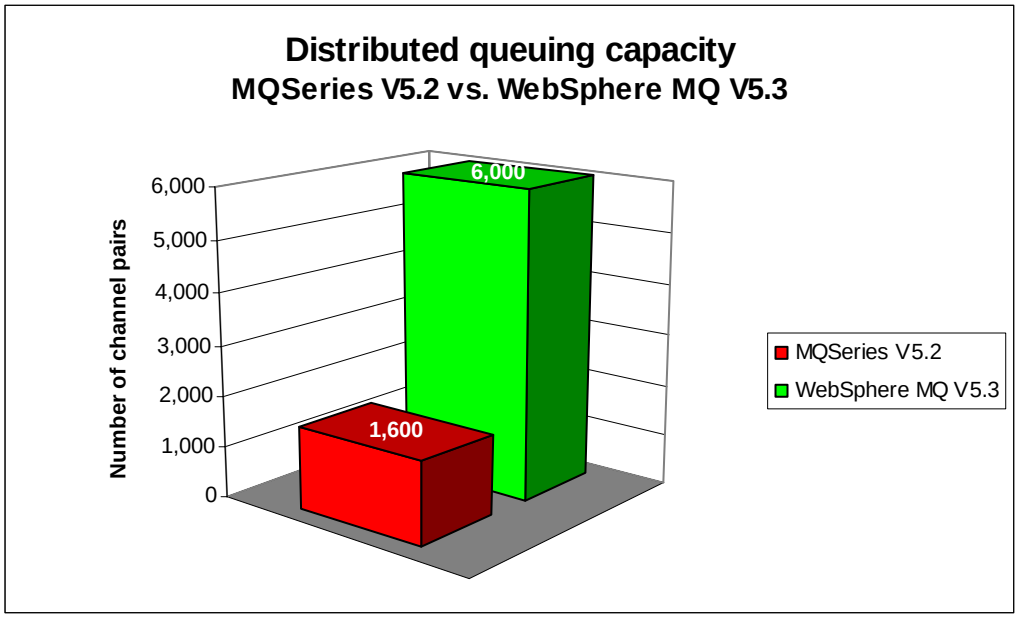

Figure 5 – Maximum number of server channels

Note: Messaging in these tests uses a rate of 1 round trip per server channel pair per minute.

2 Performance headlines

The measurements for the local queue manager scenario are for processing messages with no *think-time*. For the client channel scenario and distributed queuing scenario, there are also measurements for *rated* messaging.

No think-time is when the driving applications does not wait after getting a reply message before submitting subsequent request messages—this is also referred to as *tight-loop*.

The rated messaging tests used one round trip per driving application per *second*. In the client channel test scenarios, each driving application using a dedicated MQI-client channel, in the distributed queuing test scenarios, one or more applications submit messages over a fixed number of server channels.

All tests are automatically stopped after the response time exceeds 1 second.

2.1 Local queue manager test scenario

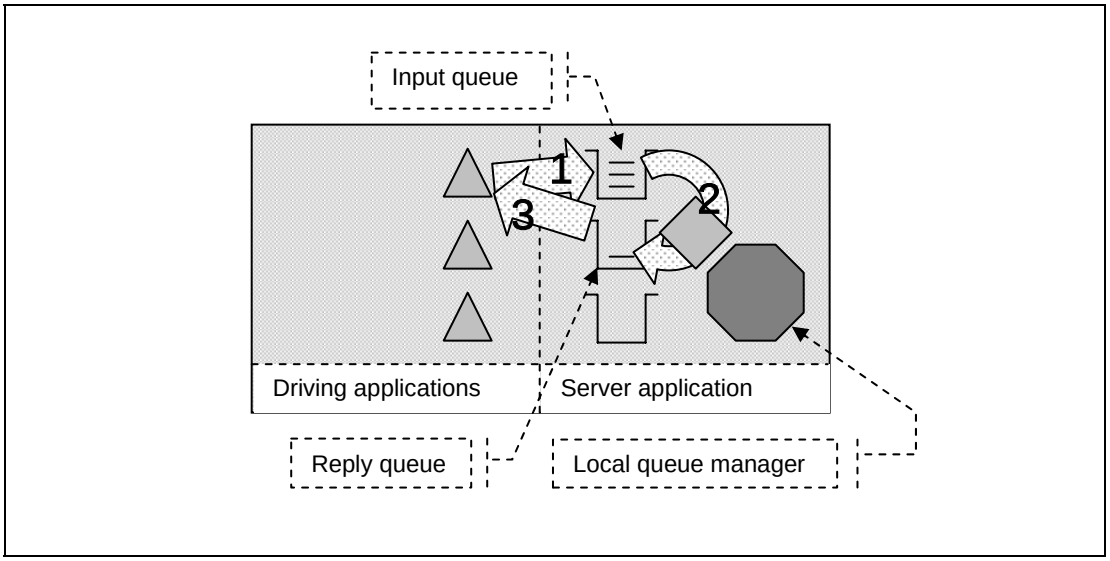

Figure 6 – Connections into a local queue manager

1) The driving application puts a message to the common input queue on the local queue manager, and holds on to the message identifier returned in the message descriptor. 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 to the common reply queue. The queue manager copies over the message identifier from the request message to the correlation identifier of the reply message.

3) The driving application gets a reply from the common reply queue using the message identifier held from when the request message was put to the common input queue, as the correlation identifier in the message descriptor.

Nonpersistent and persistent messages were used in the local queue manager tests, with a message size of 2K. The effect of message throughput with larger messages sizes is investigated in '**MQI response times: 50bytes to 2MB – local queue manager**'—**Page 17**, and '**Large messages: 20K and 200K – local queue manager** '—**Page 18**.

2.1.1 Nonpersistent messages – local queue manager

Figure 7 and **Figure 8** shows the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario (see the previous page), and WebSphere MQ V5.3 compared to Version 5.2.

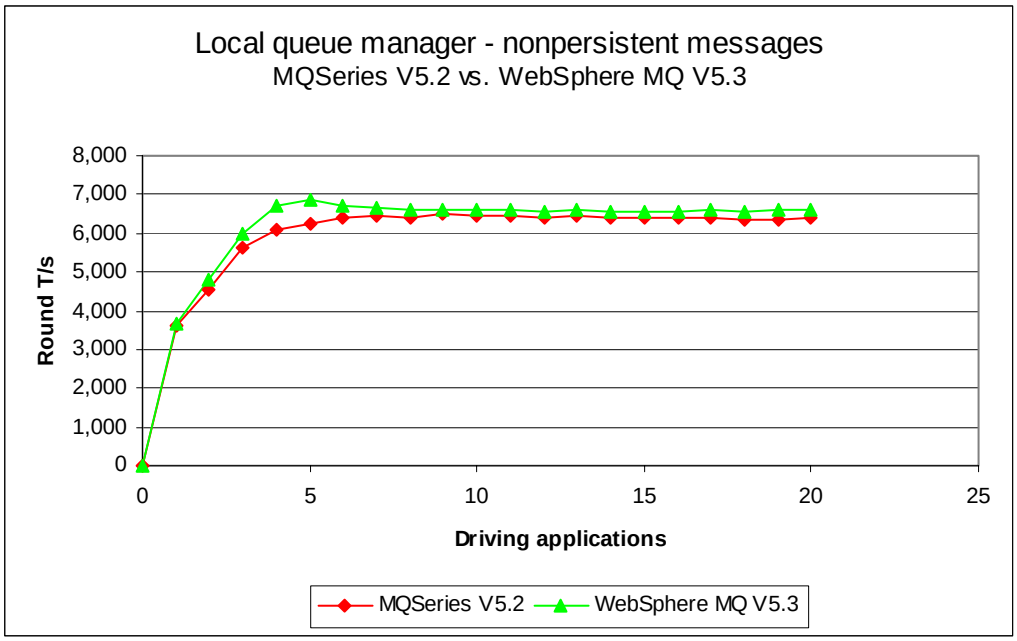

Figure 7 – Performance headline, nonpersistent messages, local queue manager

Note: Messaging in these tests is with no think-time.

Figure 7 and **Table 1** shows that the peak throughput of nonpersistent messages has increased in Version 5.3 compared to Version 5.2 by 6% (6,480 cf. – 6,481 RT/s), with the advantage of improved scalability when using as few as 4 driving applications (see chart). Using 5 driving applications nonpersistent throughput is improved by 9% (6,267 cf. 6,841 RT/s).

Test name: local_np1	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	(5) **9** (20)	(6,267) **6,480** (6,404)	n/a	(0.001) **0.002** (0.004)
WebSphere MQ V5.3	**5** (9) (20)	**6,841** (6,602) (6,588)	**+9** (+2) (+3)	**0.001** (0.002) (0.004)

Table 1 – Performance headline, nonpersistent messages, local queue manager

Note: The large bold numbers in the table above show the peak number of round trips per second, and the number of driving applications used to achieved the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V5.3 and Version 5.2.

2.1.2 Persistent messages – local queue manager

Queue manager log configuration:

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

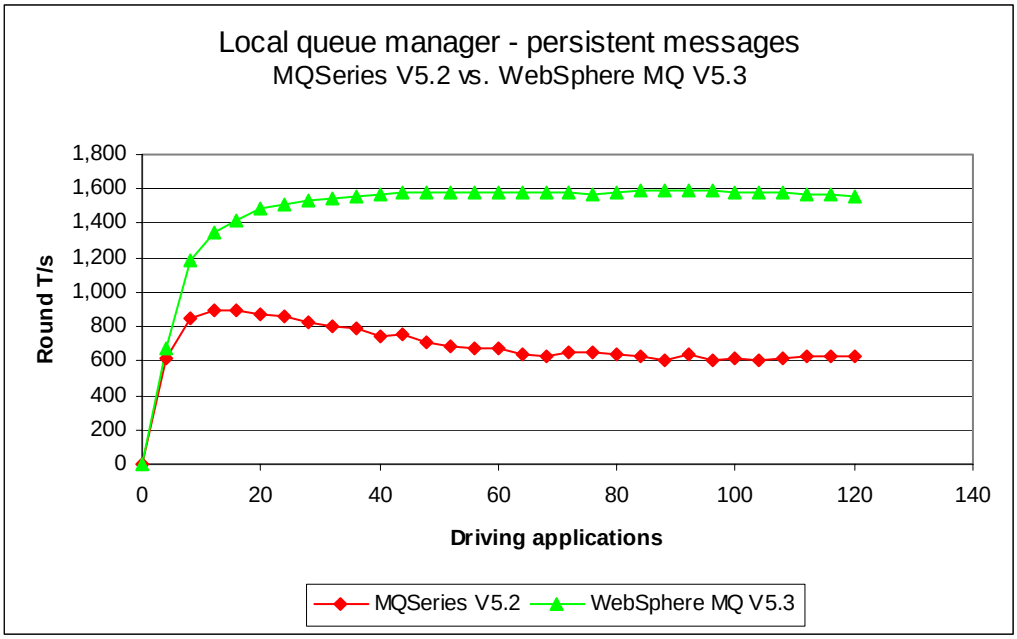

Figure 8 – Performance headline, persistent messages, local queue manager

Note: Messaging in these tests is with no think-time.

Figure 8 and **Table 2** show that the peak throughput of persistent messages has increased by 78% (899 cf. 1,596 RT/s) comparing Version 5.2 to Version 5.3. Using 96 driving applications persistent throughput is improved by 163% (607 cf. 1,596 RT/s). Version 5.3 has the advantage of improved scalability when using 8 or more driving applications.

Test name: local_pm1	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	**12** (96) (120)	**899** (607) (627)	n/a	**0.014** (0.163) (0.212)
WebSphere MQ V5.3	(12) **96** (120)	(1,346) **1,596** (1.561)	(+50) **+163** (+149)	(0.010) **0.071** (0.090)

Table 2 – Performance headline, persistent messages, local queue manager

2.2 Client channels test scenario

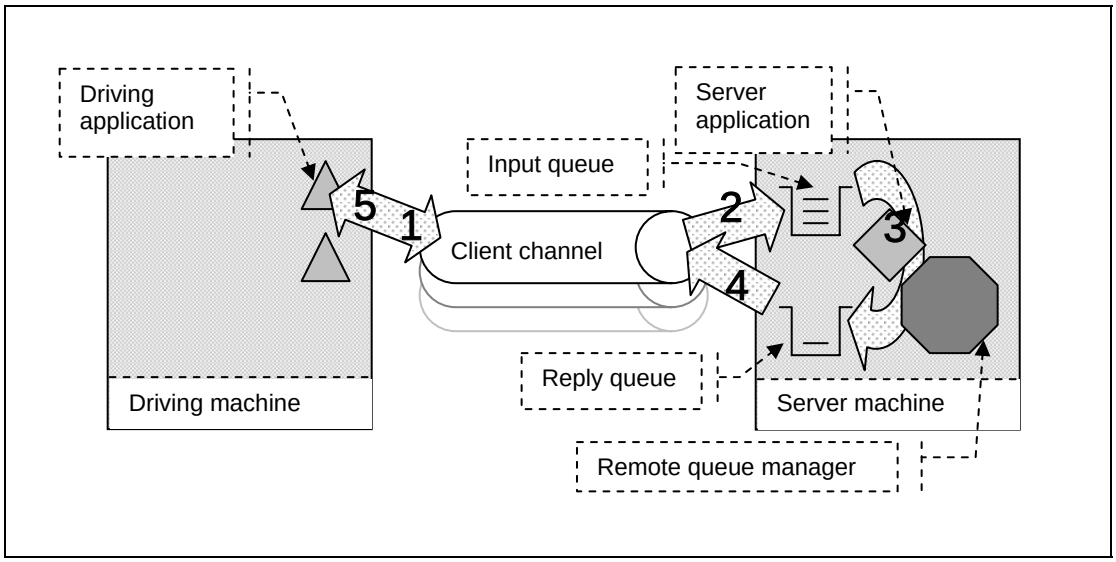

Figure 9 – MQI-client channels into a remote queue manager

1, 2) The driving application puts a request message (over a client channel), to the common input queue, and holds on to the message identifier returned in the message descriptor. 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 to the common reply queue. The queue manager copies over the message identifier from the request message to the correlation identifier of the reply message.

4, 5) The driving application gets the reply message (over the client channel), from the common reply queue. The driving application uses the message identifier held from when the request message was put to the common input queue, as the correlation identifier in the message descriptor.

Nonpersistent and persistent messages were used in the client channel tests, with a message size of 2K. The effect of message throughput with larger messages sizes is investigated in **'Large messages: 20K and 200K – client channels'** —**Page 21**.

2.2.1 Nonpersistent messages – client channels

Figure 10 and **Figure 11** shows the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client chanel scenario (see **Figure 9** above), and Version 5.2 compared with Version 5.3.

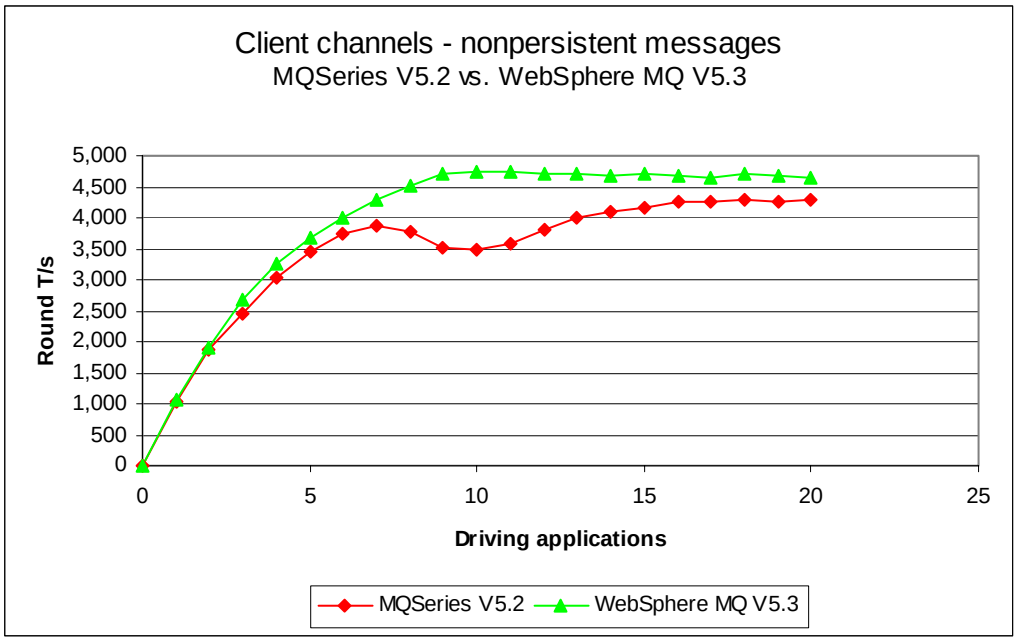

Figure 10 – Performance headline, nonpersistent messages, client channels

Note: Messaging in these tests is with no think-time

Figure 10 and **Table 3** show that the peak throughput of nonpersistent messages is up by 11% (4,293 cf. – 4,745 RT/s) comparing Version 5.2 to Version 5.3, with the advantage of improved scalability when using as few as 5 driving applications. When using as few as 10 driving applications throughput throughput is improved by 36% : (3,490 cf. 4,745 RT/s).

Test name: clnp1 (inetd)	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	(10) **18** (20)	(3,490) **4293** (4,284)	n/a	(0.003) **0.005** (0.005)
WebSphere MQ V5.3	**10** (18) (20)	**4,745** (4,694) (4,640)	**+36** (+9) (+8)	**0.002** (0.005) (0.005)

Table 3 – Performance headline, nonpersistent messages, client channels

2.2.2 Persistent messages – client channels

Queue manager log configuration:

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

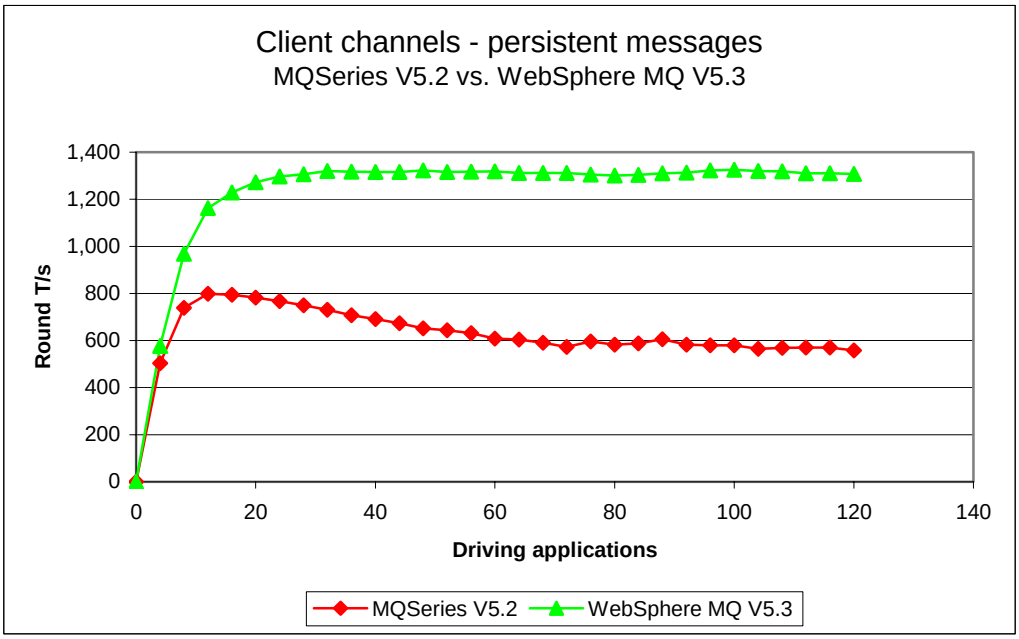

Figure 11 – Performance headline, persistent messages, client channels

Note: Messaging in these tests is with no think-time.

Figure 11 and **Table 4** show that the peak throughput of persistent messages is up by 66% (799 cf. 1,325 Round T/s) comparing Version 5.2 to Version 5.3, with the advantage of improved scalability giving the most performance improvement using 120 driving applications (throughput up by 134% : 558 cf. 1,307 RT/s).

Test name: clpm3 (inetd)	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	**12** (100) (120)	**799** (579) (558)	n/a	**0.016** (0.256) (0.344)
WebSphere MQ V5.3	(12) **100** (120)	(1,163) **1,325** (1,307)	(+46) +129 (+134)	(0.012) 0.087 (0.106)

Table 4 – Performance headline, persistent messages, client channels

2.2.3 Runmqlsr vs. inetd listener – client channels

For the following client channel measurements, the messaging rate used is 1 round trip per *second* per MQI-client channel, i.e. a request message outbound over the client channel and a reply message inbound over the channel per second. These tests also compare the difference between *inetd* channels (the 'amqcrsta' process started by inetd) with *runmqlsr* channels (the 'runmqlsr' process started by the user).

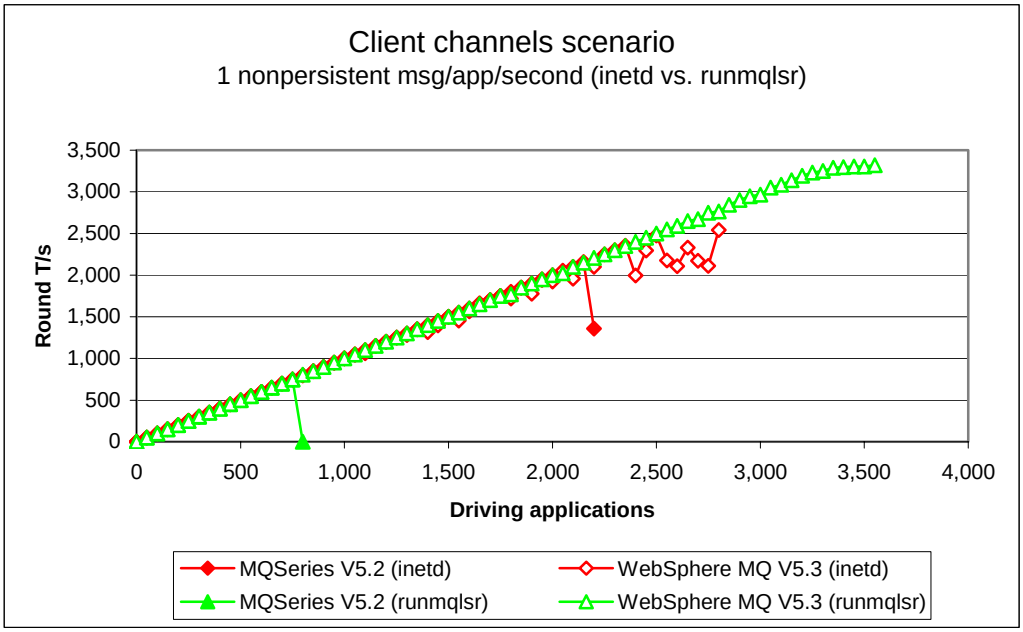

Figure 12 – Inetd vs. runmqlsr listener, client channels

Note: Messaging in these tests is 1 round trip per driving application per second.

Figure 12 and **Table 5** shows how the inetd and runmqlsr listener in WebSphere MQ V5.3 give improved scalability by permitting a larger number of MQI-client connections into a single queue manager. In Version 5.2, the 'runmqlsr' listener physically breaks (a well know and documented problem).

In Version 5.3 it is now possible to connect more than 500 driving application into a single queue manager (17,500 fastpath MQI-client connections: refer to **'Capacity limits – client channels'**—**Page 3**). Furthermore, the 'runmqlsr' has a reduced resource utilisation (one thread per connection vs. a process per connection for the 'inetd' listened and a smaller memory footprint), so is now the **preferred** method of running client channels and server channels.

Test name	Apps	Rate/app/hr	Round T/s	%	Resp time (s)
clnp1_r3600 (inetd) (MQSeries V5.2)	2,700 (2,150)	3,600	2,173 (2,155)	+1	0.947 (0.003)
clnp1_r3600_runmqlsr (MQSeries V5.2)	3,250 (750)	3,600	3,232 (750)	+331	0.867 (0.016)
clpm3_r3600 (inetd) (MQSeries V5.2)	1000 (500)	3,600	985 (500)	+97	0.824 (0.040)
clpm3_r3600_runmqlsr (MQSeries V5.2)	1,200 (550)	3,600	1,165 (192)	+507	0.939 (0.302)

Table 5 – 1 round trip per driving application per second, client channels

*Note: **Table 5** shows the response time per round trip is given closer to one second for Version 5.3. This does not show Version 5.3 to be worse than Version 5.2 for two reasons: for Version 5.3 there are more driving applications and hence a higher message throughput at the point of constraint, and the approach of the one second response time happens in a more controlled manner. As the one second response time is exceeded, two products behave as illustrated below:*

clnp1_r3600 - Version 5.2			clnp1_3600 - Version 5.3		
Apps	**Round T/s**	**Resp time (s)**	**Apps**	**Round T/s**	**Resp time (s)**
2,150	2,155	0.003	2,650	2,329	0.887
2,200	1,219	1.697	2,700	2,173	0.947
2,250	1,328	2.141	2,750	2,112	1.136

Table 6 – Driving applications vs response time

Note: Directly correlating Apps to Round T/s, with Version 5.3 the message throughput and response time peak and degrade in a more controlled manner compared to Version 5.2.

Figure 13 shows the reduced memory requirement of an MQI-client connection comparing the inetd listener to the runmqlsr listener in WebSphere MQ V5.3.

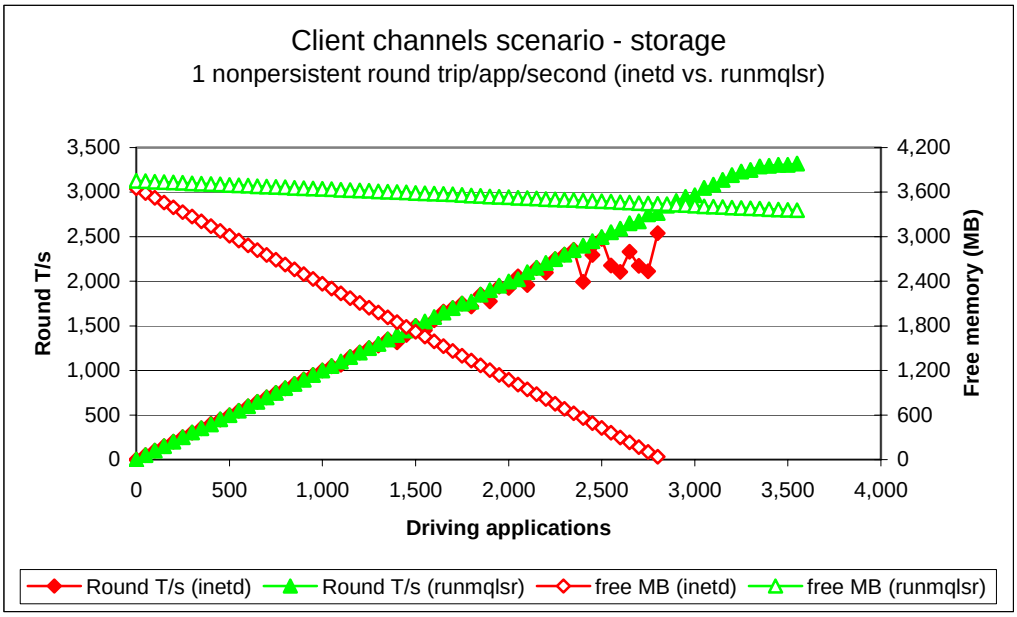

Figure 13 – Inetd vs. runmqlsr listener, free memory, client channels

Note: Messaging in these tests is 1 round trip per driving application per second

Test name	Apps	Free (MB)
clnp1_r3600 (inetd)	100 (1000)	3,524 (2,367)
clpm3_r3600_runmqlsr	100 (1000)	3,654 (3,552)

Table 7 – Inetd vs. runmqlsr listener, free memory, client channels

*Note: The free memory shown in **Table 7** represents the available real memory, not swap memory.*

For further calculations on the swap reservation and shared memory utilisation refer to **'Table 18 – Client capacity, shared memory utilisation'—Page 29**, and **'Table 20 – Distributed queuing capacity, '—Page 31**.

2.3 Distributed queuing test scenario

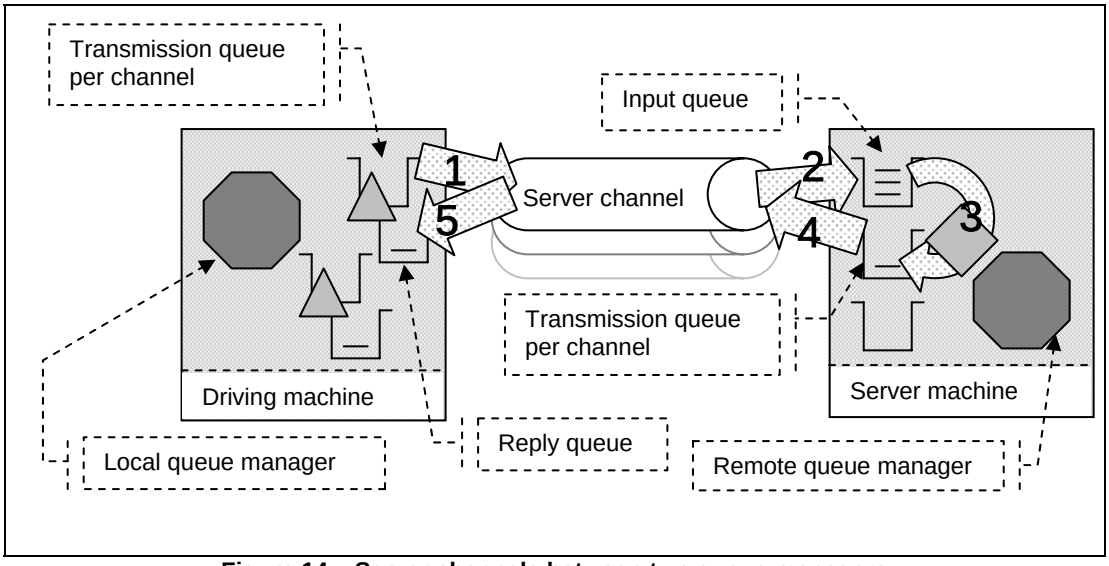

Figure 14 – Server channels between two queue managers

1) The driving application puts a message to a local definition of a remote queue located on the server machine, and holds on to the message identifier returned in the message descriptor. The driving application then waits indefinitely for a reply to arrive on a local queue.

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

3) The server application gets messages from the common input queue, and places a reply to the queue name extracted from the messages descriptor (the name of a local definition of a remote queue located on the driving machine). The queue manager copies over the message identifier from the request message to the correlation identifier of the reply message.

4) The message channel agent takes messages off the transmission queue and sends them over the channel to the driving machine.

5) The driving application gets a reply from a local queue. The driving application uses the message identifier held from when the request message was put to the local definition of the remote queue, as the correlation identifier in the message descriptor

Nonpersistent and persistent messages were used in the distributed queuing tests, with a message size of 2K. The effect of message throughput with larger messages sizes is investigated in **'Large messages: 20K and 200K – distributed queuing'**—**Page 23**.

2.3.1 Nonpersistent messages – server channels

Figure 15 and **Figure 16** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario (see **Figure 14** on the previous page above), and WebSphere MQ V5.3 compared to Version 5.2.

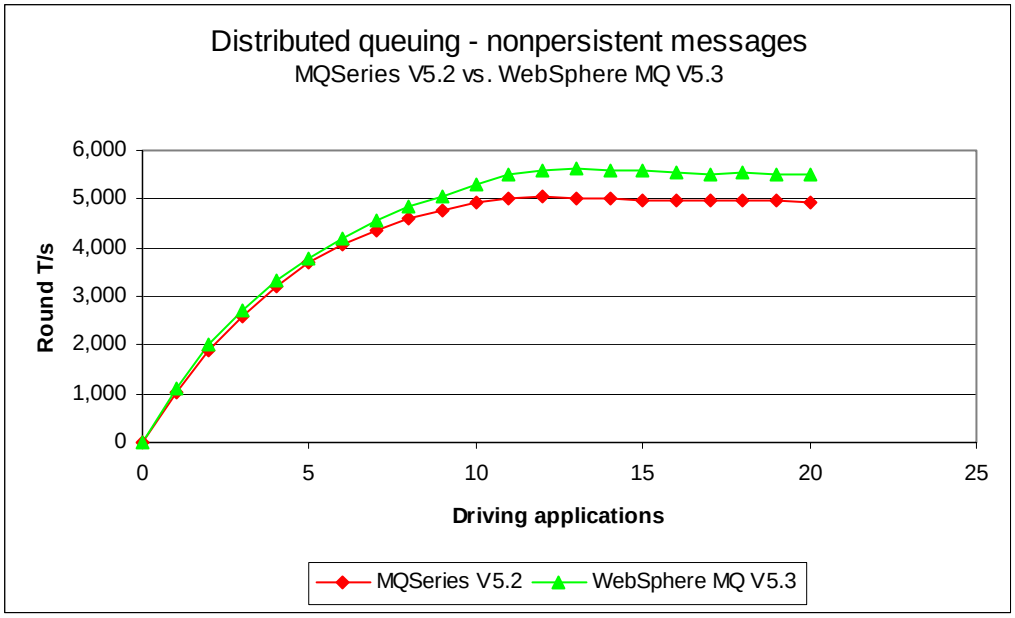

Figure 15 – Performance headline, nonpersistent messages, server channels

Note: Messaging in these tests is with no think-time.

Figure 15 and **Table 8** show that the peak throughput of nonpersistent messages has increased by 11% (5,057 cf. 5,611 RT/s) comparing Version 5.2 to Version 5.3, also with the advantage of improved scalability when using as few as 12 driving applications (throughput is up by 11% : ,5,057 cf. 5,611 RT/s). Using 20 driving applications throughput is improved by 12% : 4,935 cf. 5,514 RT/s.

Test name: dqnp1 (inetd)	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	**12** (13) (20)	**5,057** (5,032) (4,935)	n/a	**0.003** (0.003) (0.005)
WebSphere MQ V5.3	(12) **13** (20)	(5,594) **5,611** (5,514)	(+11) **+12** (+12)	(0.002) **0.002** (0.004)

Table 8 – Performance headline, nonpersistent messages, server channels

2.3.2 Persistent messages – server channels

Queue manager log configuration:

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

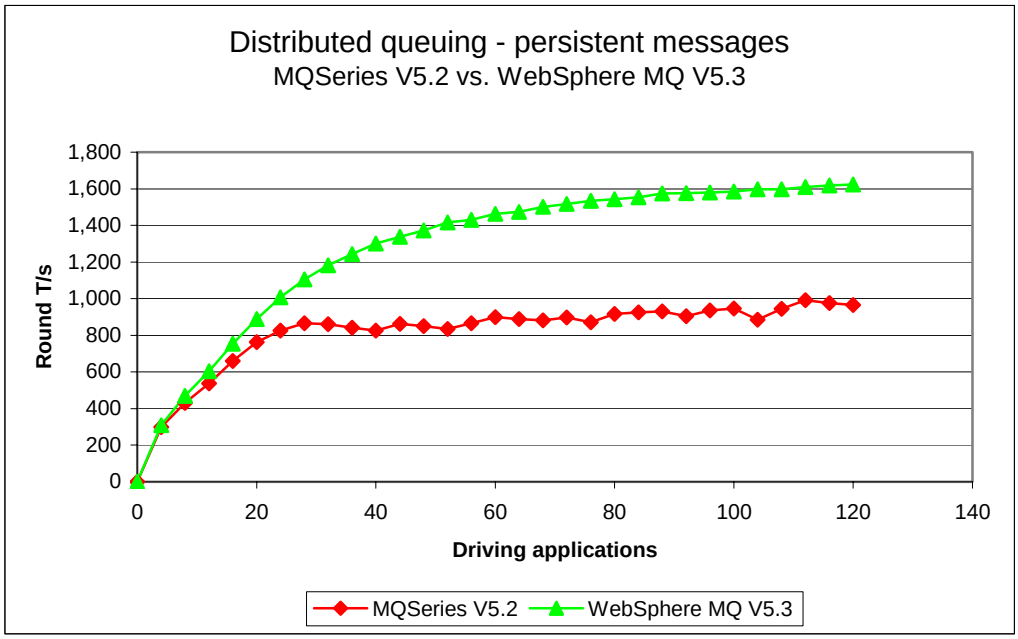

Figure 16 – Performance headline, persistent messages, server channels

Note: Messaging in these tests is with no think-time

Figure 16 and **Table 9** show that the peak throughput of persistent messages has increased by 64% (992 cf. 1,623 RT/s) comparing Version 5.2 to Version 5.3, also with the advantage of improved scalability after 20 driving applications.

The persistent message tests in **Figure 16** do not constrain on response time. This shows that in the distributed queuing scenario used for the tests in this section (using 2 server channel pairs between 2 queue managers running on separate server machines), the 2 queue managers can maintain a message throughput of at least 1,623 round trips per second.

Test name: dqpm1 (inetd)	Apps	Round T/s	%	Resp time (s)
MQSeries V5.2	**112** (120)	**992** (965)	n/a	**0.125** (0.135)
WebSphere MQ V5.3	(112) **120**	(1,609) **1,623**	(+62) **+68**	(0.082) **0.093**

Table 9 – Performance headline, persistent messages, server channels

2.3.3 Runmqlsr vs. inetd listener – server channels

For the following distributed queuing measurements, the messaging rate used is 1 round trip per driving application per *second*, i.e. a request message outbound over the sender channel, and a reply message inbound over the receiver channel per second. Note that there are a fixed number of 4 server channel pairs for the nonpersistent messaging tests, and 2 pairs for the persistent message tests. These tests also compare the difference between *inetd* channels (the 'amqcrsta' process started by inetd, and the 'runmqchl' process started by the queue manager) with *runmqlsr* channels (the 'runmqlsr' process started by the user, and the 'runmqchi' process started with the queue manager).

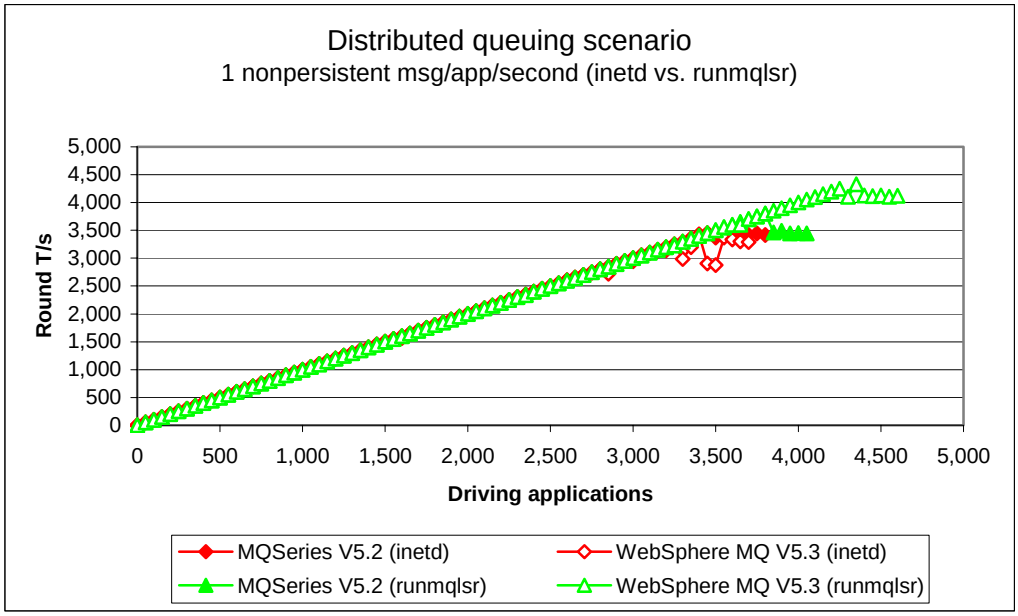

Figure 17 – Inetd vs. runmqlsr listener, server channels

Note: Messaging in these tests is 1 round trip per driving application per second

Figure 17 shows that there is little difference between the inetd listener and runmqlsr in terms of the number of round trips that can be achieved per second before the round trip response time exceeds 1 second.

Test name	Apps	Rate/app/hr	Round T/s	%	Resp time
dqnp1_r3600 (inetd) MQSeries V5.2	3,400 (3,450)	3,600	3,430 (3,449)	-1	0.091 (0.005)
dqnp1_r3600_runmqlsr MQSeries V5.2	4,350 (3,800)	3,600	4,331 (3,803)	+14	0.329 (0.091)
dqpm1_r3600 (inetd) MQSeries V5.2	1,400 (1,150)	3,600	1,383 (1,037)	+33	0.190 (0.680)
dqpm1_r3600_runmqlsr MQSeries V5.2	1,700 (1,200)	3,600	1,696 (1,080)	+57	0.434 (0.935)

Table 10 – 1 round trip per driving application per second, server channels

Note: It was anticipated that the WebSphere MQ V5.3 runmqlsr should have a reduced resource utilisation compared to the WebSphere MQ V5.3 inetd listener and the Version 5.2 listeners; therefore a larger number of driving applications would be able to process more messages over the server channels before the measurement constrained. However, analysis of the performance data revealed that a number of messages were building up on the input queue because the server application threads were not able to remove and process them quickly enough (there were spare CPU cycles because the system processors were being utilised to less than 60%). Uusing more threads in the server program or increasing the size of the nonpersistent queue buffer is not likely to provide a performance enhancement, whereas, using more input queues is most likely to give a higher message throughput with less than one second response time.

3 Large messages

All tests are automatically stopped after the response time of 1 round trip exceeds 1 second.

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

Queue manager log configuration:

```
LogPrimaryFiles=3, LogFilePages=2048, LogBufferPages=17
```

Figure 18 and **Figure 19** below show that the response for MQPUT/GET pairs is improved for all persistent message sizes between 50bytes and 2MB. For nonpersistent messages the response time for MQPUT/GET pairs is improved for messages sizes above 256KB.

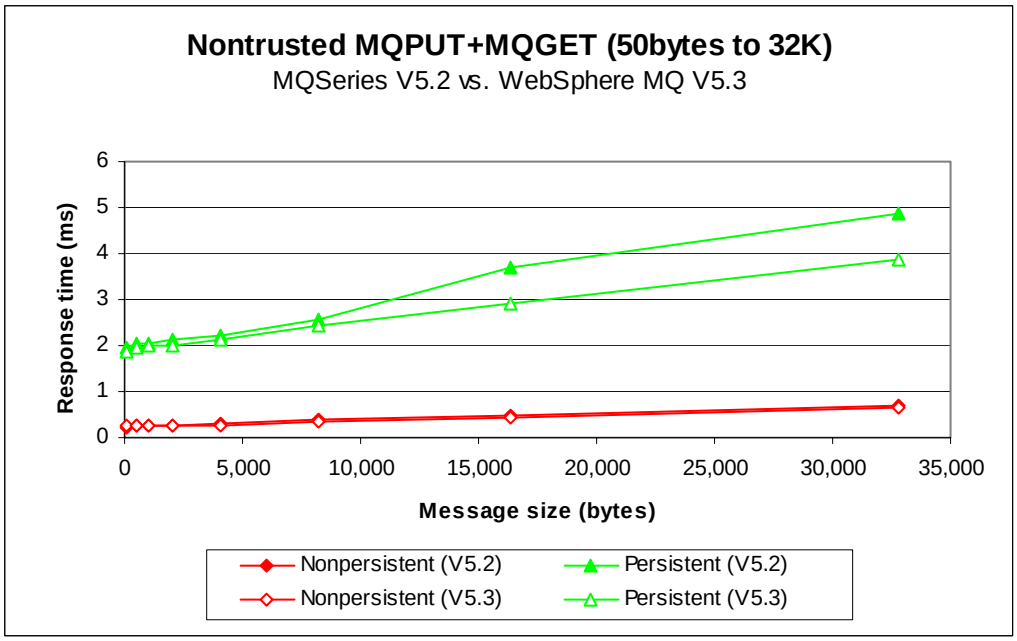

Figure 18 – The effect of message size on MQI response time (50bytes to 32K)

Note: Messaging in these tests is with no think-time.

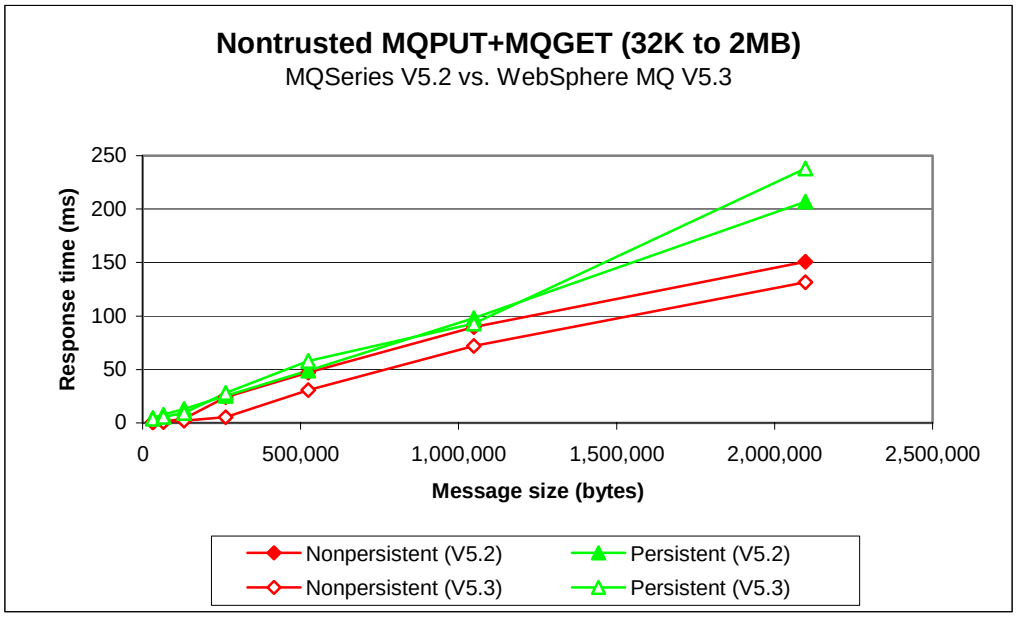

Figure 19 – The effect of message size on MQI response time (32K to 2MB)

Note: Messaging in these tests is with no think-time.

3.2 Large messages: 20K and 200K – local queue manager

Queue manager log configuration for persistent message tests:

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

Test name	Apps	Msg size	Round T/s	%	Resp time (s)
local_np1_2K (MQSeries V5.2)	5 (9)	2K	6,841 (6,480)	+6	0.001 (0.002)
local_np1_20K (MQSeries V5.2)	4 (4)	20K	3,868 (3,213)	+20	0.001 (0.001)
local_np1_200K (MQSeries V5.2)	3 (6)	200K	452 (358)	+26	0.008 (0.019)
local_pm3_2K (MQSeries V5.2)	96 (12)	2K	1,596 (899)	+78	0.071 (0.014)
local_pm3_20K (MQSeries V5.2)	20 (12)	20K	460 (426)	+8	0.048 (0.032)
local_pm3_200K (MQSeries V5.2)	4 (4)	200K	33 (32)	+2	0.122 (0.125)

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

Note: Messaging in these tests is with no think-time.

Figure 20 below shows how the throughput of both small nonpersistent and persistent messages is improved slightly in WebSphere MQ V5.3, using as few as 3 driving applications.

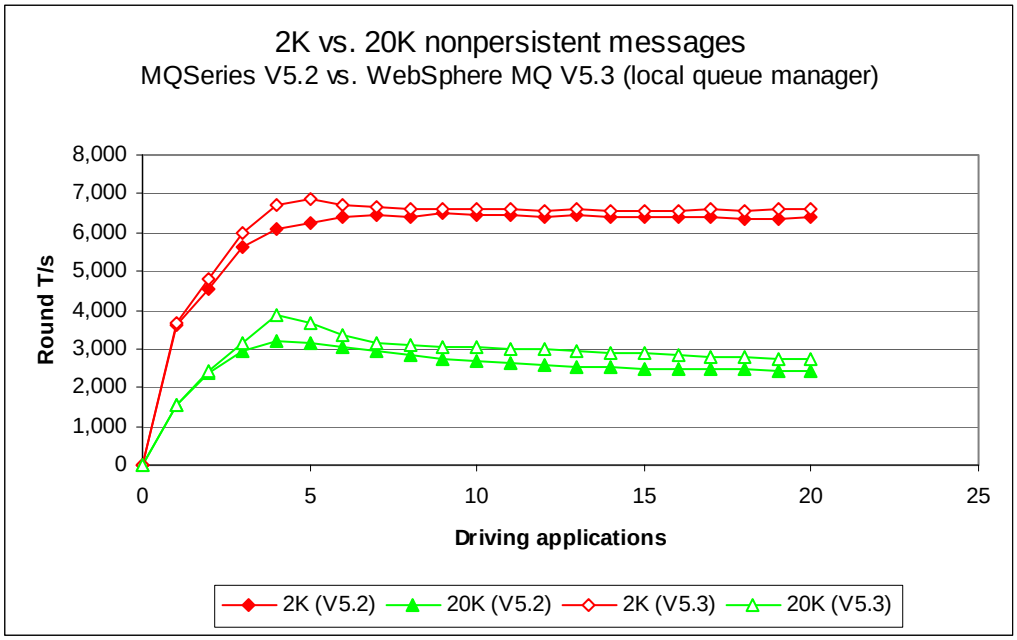

Figure 20 – 2K and 20K nonpersistent messages, local queue manager

Figure 21 shows how the throughput of 2K persistent messages is improved significantly using as few as 8 applications. The throughput of 20K persistent messages is improved slightly, in WebSphere MQ V5.3, using more than 20 driving applications.

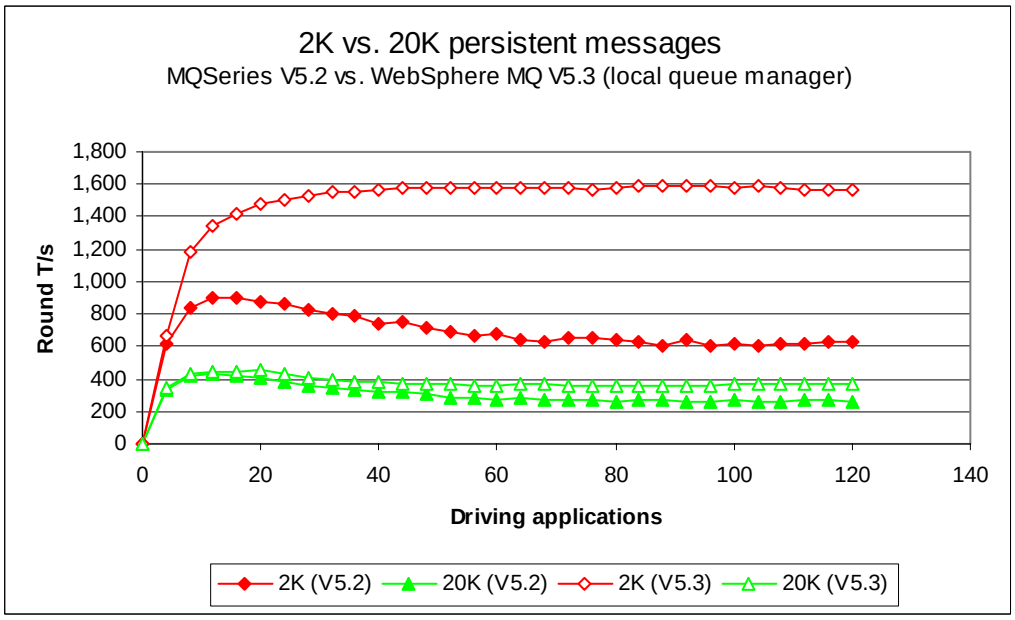

Figure 21 – 2K and 20K persistent messages, local queue manager

Figure 22 shows how the throughput of 200K nonpersistent messages is improved slightly.

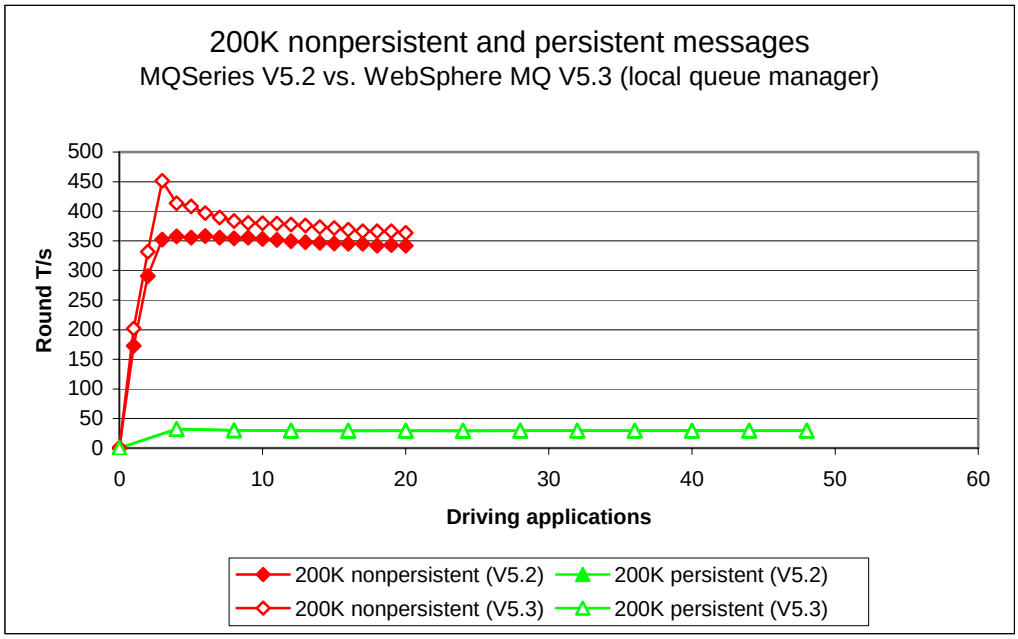

Figure 22 – 200K nonpersistent and persistent messages, local queue manager

3.3 Large messages: 20K and 200K – client channels

Queue manager log configuration for persistent message tests:

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

Test name	Apps	Msg size	Round T/s	%	Resp time (s)
clnp1_2K (MQSeries V5.2)	10 (18)	2K	4,745 (4,293)	+11	0.002 (0.005)
clnp1_20K (MQSeries V5.2)	20 (15)	20K	1,522 (1,483)	+3	0.015 (0.012)
clnp1_200K (MQSeries V5.2)	20 (20)	200K	172 (172)	+0	0.138 (0.139)
clpm3_2K (MQSeries V5.2)	100 (12)	2K	1,325 (799)	+66	0.032 (0.016)
clpm3_20K (MQSeries V5.2)	16 (16)	20K	428 (400)	+7	0.041 (0.042)
clpm3_200K (MQSeries V5.2)	4 (4)	200K	33 (33)	+0	0.117 (0.120)

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

Note: Messaging in these tests is with no think-time, and the inetd 'amqcrsta' listener.

Figure 23 shows how the throughput of small nonpersistent messages is improved slightly in WebSphere MQ V5.3, using as few as 3 driving applications.

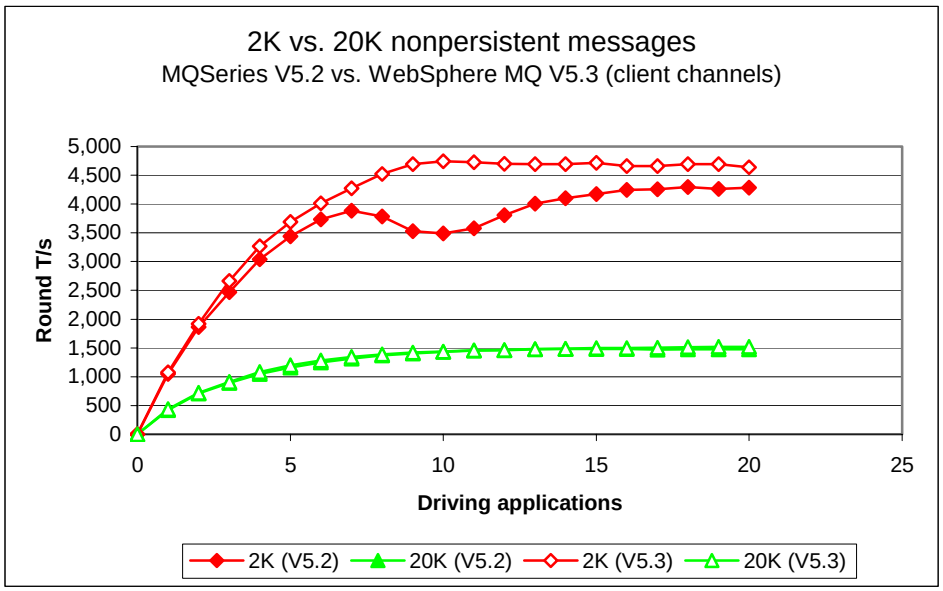

Figure 23 – 2K and 20K nonpersistent messages, client channels

Figure 24 shows how the throughput of 2K persistent messages is improved significantly. The throughput of 20K persistent messages is improved slightly, in WebSphere MQ V5.3, using more than 20 driving applications. Using 60 or more driving applications Version 5.3 gives a maintained persistent throughput of 350 round trips or more per second (5% more than Version 5.2).

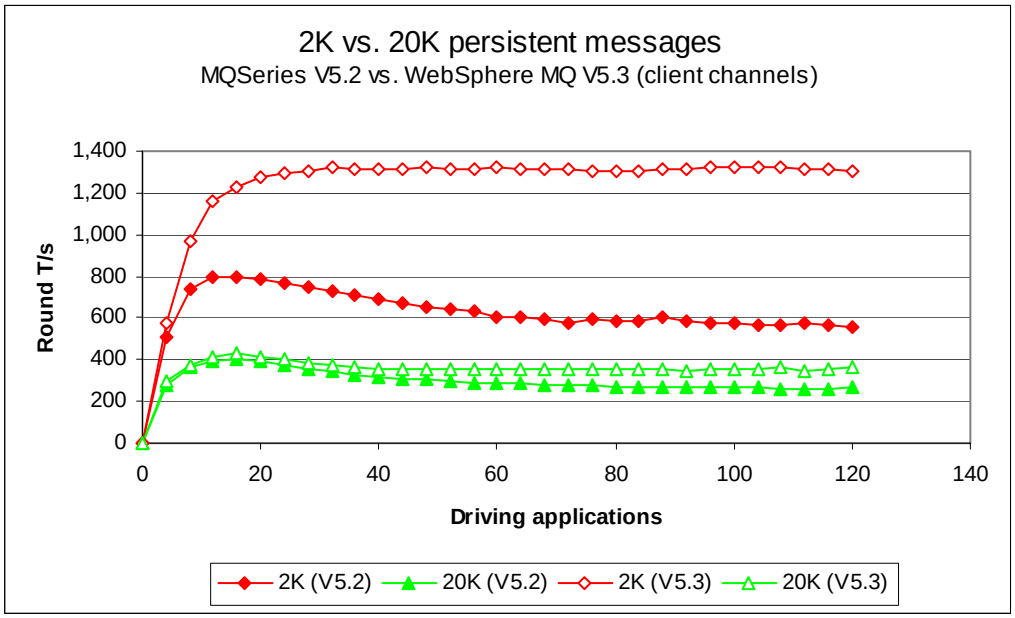

Figure 24 – 2K and 20K persistent messages, client channels

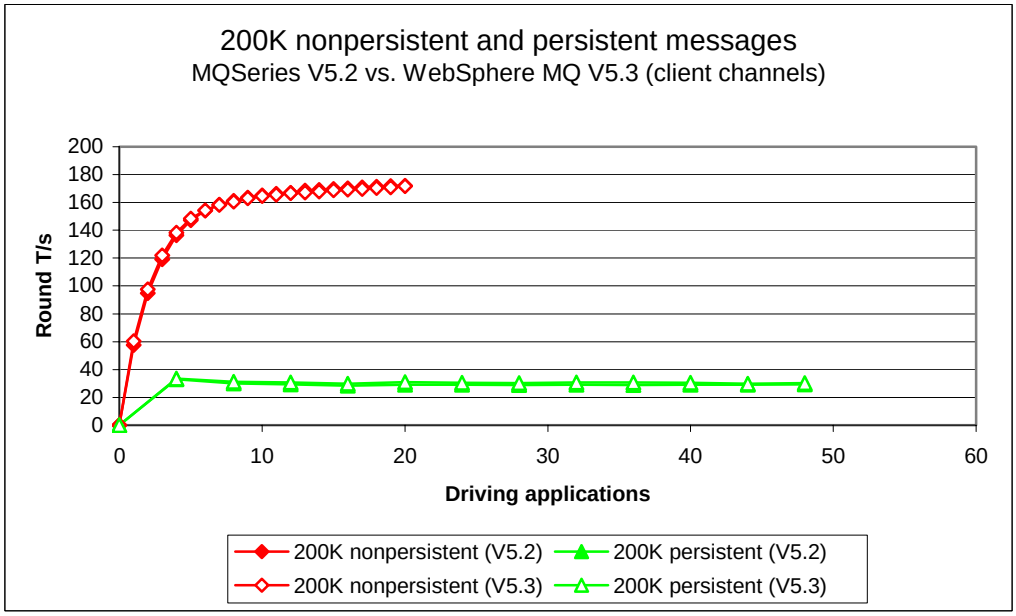

Figure 25 – 200K nonpersistent and persistent messages, client channels

3.4 Large messages: 20K and 200K – distributed queuing

Queue manager log configuration for persistent message tests:

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

Test name	Apps	Msg size	Round T/s	%	Resp time (s)
dqnp1_2K (MQSeries V5.2)	13 (12)	2K	5,611 (5,057)	+11	0.002 (0.003)
dqnp1_20K (MQSeries V5.2)	15 (16)	20K	1,601 (1,615)	-1	0.011 (0.012)
dqnp1_200K (MQSeries V5.2)	12 (16)	200K	156 (165)	-6	0.090 (0.117)
dqpm1_2K (MQSeries V5.2)	120 (112)	2K	1,623 (992)	+64	0.093 (0.125)
dqpm1_20K (MQSeries V5.2)	20 (16)	20K	320 (304)	+5	0.071 (0.059)
dqpm1_200K (MQSeries V5.2)	28 (4)	200K	34 (34)	+0	0.953 (0.117)

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

Note: Messaging in these tests is with no think-time, and the inetd 'amqcrsta' listener.

Note: The measurements for 20K persistent messages show that there is little difference in the performance of large message between Version 5.2 and Version 5.3—this is because most of the time taken by the queue manager is in logging the messages to disk.

Figure 26 shows how the throughput of small nonpersistent messages is improved slightly in WebSphere MQ V5.3, using as few as five driving applications.

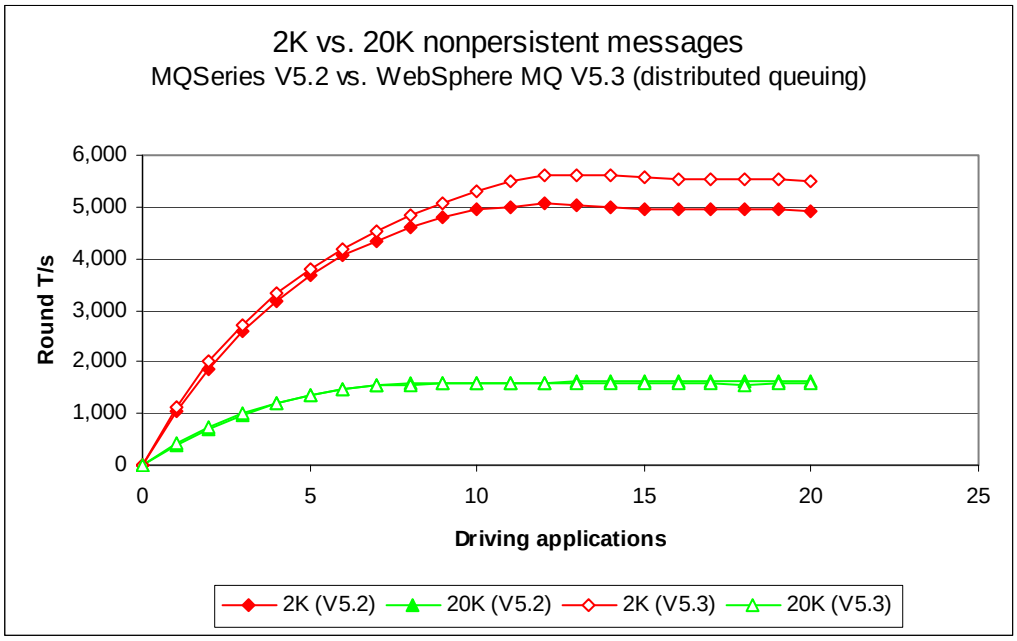

Figure 26 – 2K and 20K nonpersistent messages, server channels

Figure 27 shows how the throughput of 2K persistent messages is improved significantly (64%), and the throughput of 20K persistent messages is improved slightly, in Version 5.3 using more than 20 driving applications (5%).

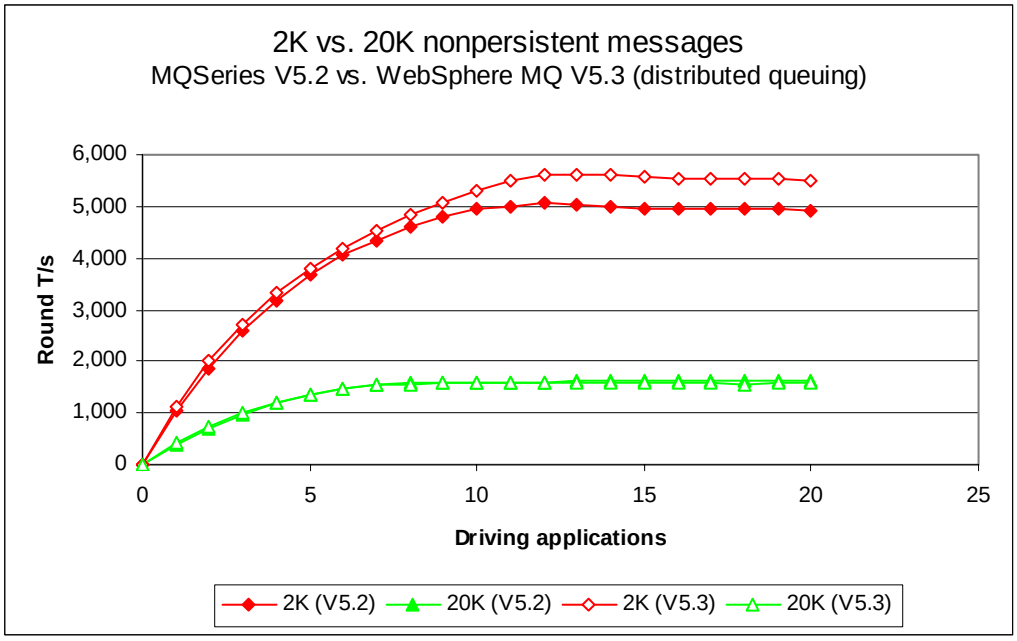

Figure 27 – 2K and 20K persistent messages, server channels

The 2K persistent message tests in **Figure 27** do not constrain on response time. Refer to **'Persistent messages – server channels'** starting on **page 15** for discussion.

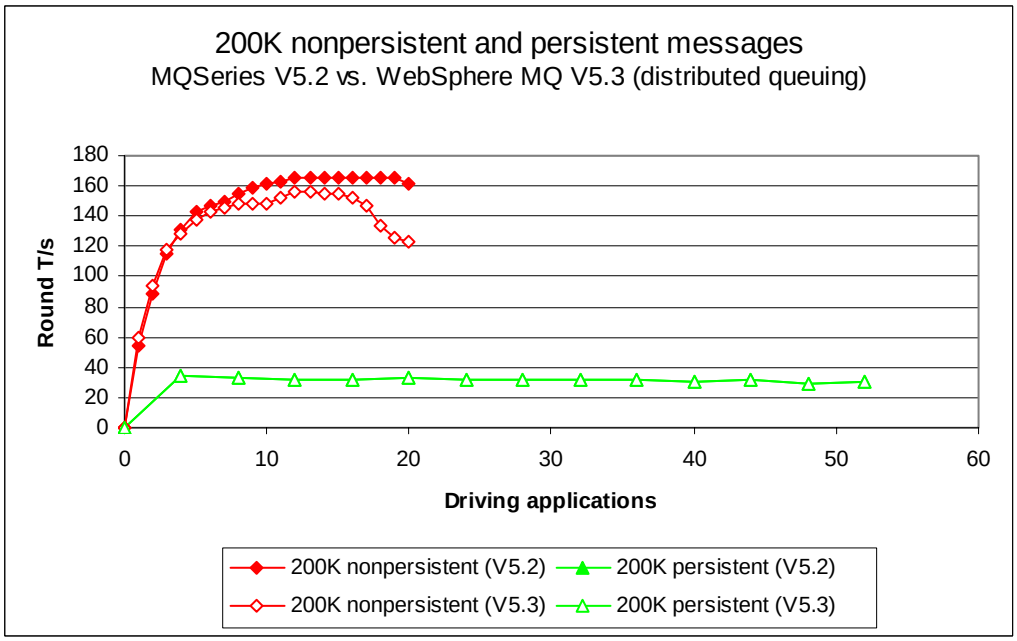

Figure 28 – 200K nonpersistent and persistent messages, server channels

With reference to **Figure 28**, the 200K nonpersistent message tests were designed to finish at 20 driving applications. The 200K persistent message tests were designed to finish at 120 driving application.

4 Trusted server application

Queue manager log configuration for persistent message tests:

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

Test name: local_np1_trusted	Apps	Round T/s	%	Resp time
MQSeries V5.2	2 (5) (20)	9,380 (7,395) (7,973)	n/a	0.000 (0.002) (0.004)
WebSphere MQ V5.3	(2) 5 (20)	(9,685) 11,395 (7,973)	(+3%) +54% (+17%)	(0.000) 0.001 (0.003)

Table 14 –Trusted server application, local queue manager

*Note: Messaging in these tests is with no think-time. See **Table 1** for nontrusted values.*

Figure 29, Table 14 and **Table 15** shows that the peak throughput of trusted messages has increased in Version 5.3 compared to Version 5.2 by 21% (9,380 cf. – 11,395 RT/s), with the advantage of improved scalability when using as few as 5 driving applications . Using 5 driving applications trusted throughput is improved by 54% (7,385 cf. 11,395 RT/s).

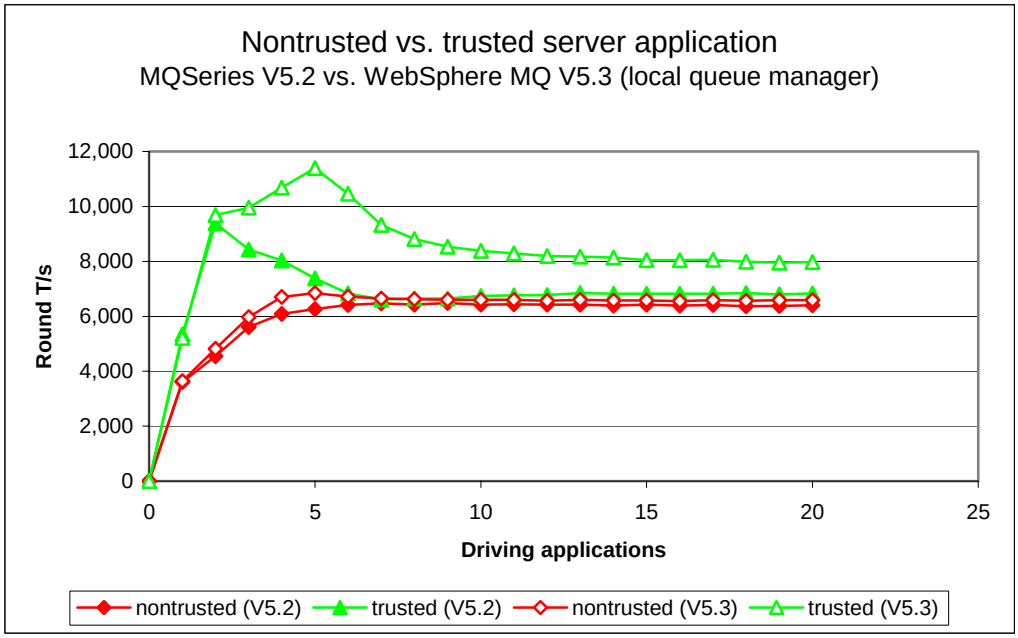

Figure 29 – Trusted server application, local queue manager

Test name	Apps	Msg size	Round T/s	%	Resp time
clnp1_trusted (MQSeries V5.2)	12 (8)	2K	5,352 (4,605)	+16	0.002 (0.002)
clpm3_trusted (MQSeries V5.2)	40 (12)	2K	1,425 (848)	+68	0.032 (0.016)
dqnp1_trusted (MQSeries V5.2)	20 (12)	2K	6,329 (5,835)	+8	0.002 (0.002)
dqpm1_trusted (MQSeries V5.2)	116 (116)	2K	1,681 (975)	+72	0.082 (0.126)

Table 15 – Trusted server application, client channels and server channels

*Note: Messaging in these tests is with no think-time, and the inetd 'amqcrsta' listener. See **Table 3** & **Table 4** for the client channel values and see **Table 8** & **Table 9** for distributed queuing values.*

5 Short sessions

A short session is a term used to describe the behaviour of an MQI application as it processes a small number of messages using one or more queues and a queue manager. The measurements in this document use an MQI-client application and the following sequence:

- 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

"Why measure short sessions?"

For each new connecting application or disconnecting application, the queue manager and Operating System must start a new process or thread and set up the new connection. As the number of connecting and disconnecting applications increases, the Operating System and queue manager are subjected to a higher load. While these requests are being serviced the queue manager has less time available to process messages, so fewer driving applications can be reconnected to the queue manager per second before the response time exceeds one second.

This effect is greater than that of reducing the total messaging throughput of the queue manager by connecting thousands of MQI applications to the queue manager (refer to **Figure 30** for an illustration).

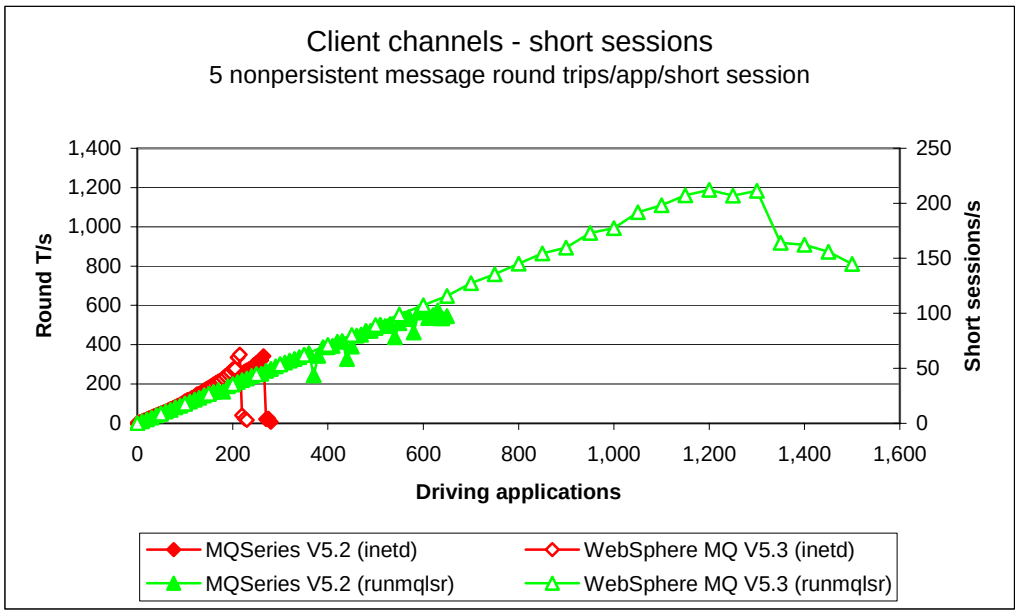

Figure 30 – Short sessions, inetd vs. runmqlsr listener, client channels

Test name	Apps	Short sessions/s	Resp time (s)
clnp1_ss_r3600 (inetd) MQSeries V5.2	205 (265)	278 (342)	0.882 (0.558)
clnp1_ss_r3600_runmqlsr MQSeries V5.2	1,200 (630)	1,189 (575)	0.422 (0.064)

Table 16 – Short sessions, nonpersistent messages, client channels

*Note: Messaging in these tests is 1 round trip per driving application per second, i.e. 1 short session per driving application every **5** seconds*

Note: The large figures in Table 16 are for WebSphere MQ V5.3 with a round trip response time of less than one second. The smaller figures in brackets are for Version 5.2. Since there are 5 round trips per short session, when the round trip response time approaches a second, the short session elapsed time will be approaching 5 seconds.

The 'runmqlsr' has a much smaller overhead of connecting to and disconnecting from the queue manager because it only uses a single thread per connection rather than an entire process. Furthermore, in Version 5.3 the maximum number of connections into a single 'runmqlsr' listener has been significantly increased making it the **preferred** method of running short sessions over client channels.

6 Performance and capacity limits

6.1 Client channels – capacity measurements

The measurements in this section are intended to test the maximum number of server channel pairs between two queue managers with a messaging rate of 1 round trip per server channel per *minute*. In previous SupportPacs, the rate used in capacity limit tests was 1 round trip per *hour*. For the same number of server channel pairs, a faster message rate gives a higher total message throughput over each channel pair. This information is intended to be more useful to the reader and assist them in projecting the results in this section to similar scenarios.

Queue manager configuration for client channels capacity tests:

```
MaxChannels=50000
```

Test name	Apps	Rate/app/hr	Round T/s	Resp time (s)	%cpu
clnp1 (inetd) (MQSeries V5.2)	10 (18)	n/a*	4,745 (4,293)	0.002 (0.005)	98 (98)
clnp1_r3600 (inetd) (MQSeries V5.2)	2,800 (2,150)	3,600	2,540 (2,155)	1.297 (0.003)	87 (69)
Clnp1_c6000_t10 (inetd)	6,000	250	410	0.843	40
Clnp1_c6000_runmqlsr_t10	6,000	1,160	1,881	0.545	73
clnp1_cmax_t10 (inetd) (MQSeries V5.2)	6,900 (6,500)	60	115 (154)	0.160 (0.145)	19 (11)
clnp1_cmax_runmqlsr_t10 (unique reply queue per app)	34,500 (21,100)	60	530 (364)	0.858 (2.422)	75 (91)

Table 17 – Capacity measurements, client channels

* Since there are 4,745 Round T/s and 3,600 seconds in one hour, the derived throughput volume per hour is calculated to be 4,745 x 3,600 = 17,082,961 Round T/hr. The reader should note that the number of 17,082,961 round trips in one hour was not physically measured over the period of one hour, however, the rates of 3,600, 250, and1,160 were actual rates set for the measurement and obtained by the queue manager system.

The effect of the number of client channels on maximum message throughput can be seen in **Figure 31** below.

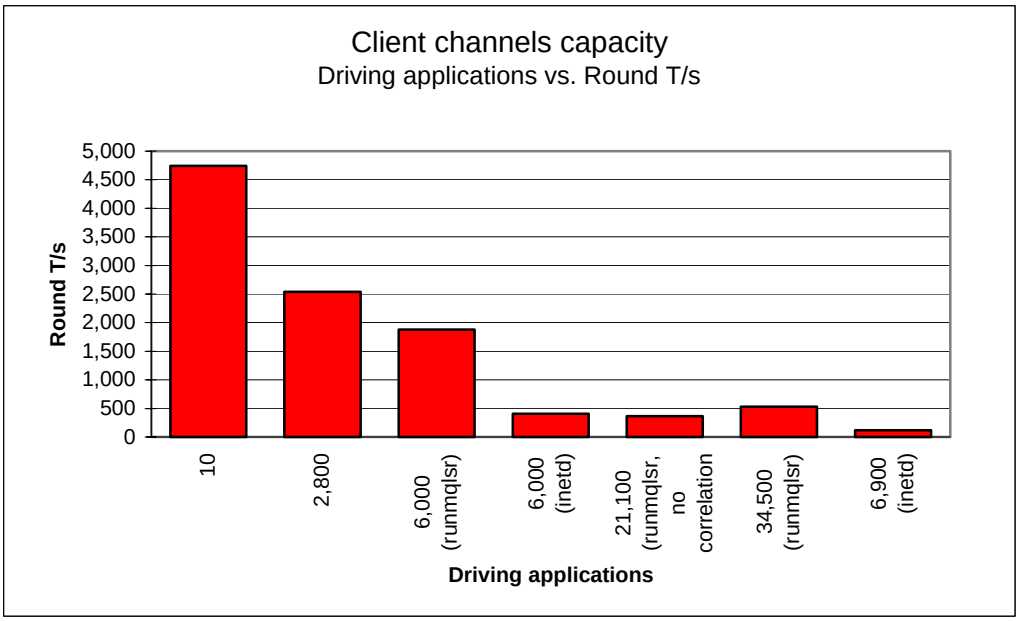

Figure 31 – Effect of number of client channels on round trips

Note: Tight-loop 1st column, fixed rate 2nd, increasing rate 3rd and 4th columns, fixed rate 5th, 6th, and 7th.

Test name	Apps	Shm	Free memory
clnp1_cmax_t10 (inetd)	6,900 (1,000)	137MB (20K/App) (92K/App)	1MB (563K/App) (1,352K/App)
clnp1_cmax_runmqlsr_t10	34,500 (1,000)	477MB (14K/App) (91K/App)	412MB (100K/App) (109K/App)
clnp1_cmax_runmqlsr_t10_no_correlid	21,100 (1,000)	1.2GB (23K/App) (113K/App)	748MB (144K/App) (143K/App)

Table 18 – Client capacity, shared memory utilisation

Note: The large figures in the table above show the shared memory and free memory measured at the given number of driving applications. The large and small figures in brackets are the proportionate shared memory and free memory cost per driving application (in this test scenario this relates to the cost of an MQI-client connection on the server machine).

6.2 Distributed queuing – capacity measurements

The measurements in this section are intended to test the maximum number of server channel pairs between two queue managers with a messaging rate of 1 round trip per server channel per *minute*. In previous SupportPacs, the rate used in capacity limit tests was 1 round trip per *hour*. For the same number of server channel pairs, a faster message rate gives a higher total message throughput over each channel pair. This information is intended to be more useful to the reader and assist them in projecting the results in this section to similar scenarios.

Queue manager and log configuration for distributed queuing capacity tests:

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

Note: The large log capacity for this test is for writing the object definitions to the log disk (the transmission queue definitions for both sides of the server channel pair, and reply queue per receiver channel on the driving machine).

Test name	Apps	Rate/app/hr	Round T/s	Resp time (s)	%cpu
dqnp1 (MQSeries V5.2)	13 (12)	n/a*	5,611 (5,057)	0.002 (0.003)	**97** **(98)**
dqnp1_r3600 (inetd) (MQSeries V5.2)	3,400 (3,450)	3,600	3,430 (3,449)	0.091 (0.005)	**64** **(78)**
dqnp1_q1000 (inetd)	1,000	3,600	1,000	0.003	**29**
dqnp1_q1000_runmqlsr	1,000	3,600	1,000	0.002	**30**
dqnp1_qmax (inetd) (MQSeries V5.2)	5,000 (2,060)	60	57 (100)	0.251 (0.3.469)	**5** **(3)**
dqnp1_qmax_runmqlsr	6,000	60	100	0.662	**9**

Table 19 – Capacity measurements, server channels

* Since there are 5,611 Round T/s, and 3,600 seconds in one hour, the derived throughput volume per hour is calculated to be 5,611 x 3,600 = 20,199,600 Round T/hr. The reader should note that the number of 20,199,600 round trips in one hour was not physically measured over the period of one hour, however, the rates of 3,600, were actual rates set for the measurement and obtained by the queue manager system.

The effect of the number of server channel pairs on maximum message throughput can be seen in **Figure 32** on the next page.

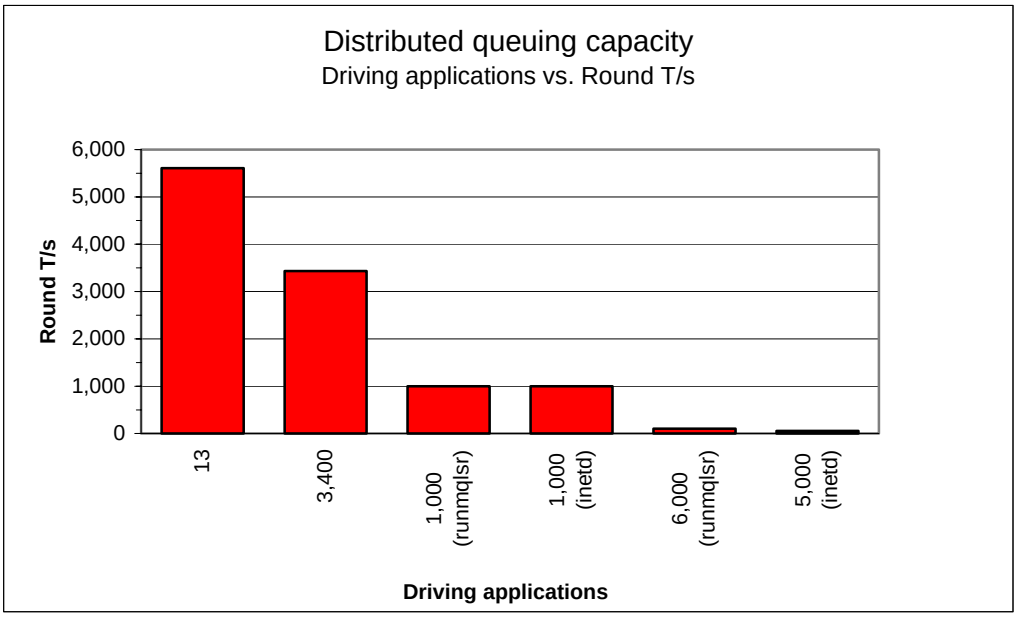

Figure 32 – Effect of number of server channels on round trips

Note: Tight-loop 1st column, fixed rate 2nd, increasing rate 3rd and 4th columns, fixed rate 5th, and 6th.

Test name	Apps	shm	Free memory
dqnp1_qmax (inetd)	5,000 (1,000)	310MB (64K/App) (185K/App)	68MB (709K/App) (170K/App)
dqnp1_qmax_runmqlsr	6,000 (1,000)	586MB (100K/App) (298K/App)	515MB (483K/App) (402K/App)

Table 20 – Distributed queuing capacity, shared memory utilisation

Note: The large figures in the table above show the shared memory and free memory measured at the given number of driving applications. The large and small figures in brackets are the proportionate shared memory and free memory cost per driving application (in this test scenario this relates to the cost of a server channel pair on the server machine).

Note: The inetd tests does not constrain on response time.

7 Tuning recommendations

7.1 Tuning the queue manager

This section highlights the tuning activities that are known to give performance benefits for WebSphere MQ V5.3; these can be applied equally to Version 5.2. The reader should note that the following tuning recommendations **may not** necessarily **need** to be applied, especially if the message throughput and/or response time of the queue manager system already meets the required level. Some tuning recommendations that follow may degrade the performance of a previously balanced system if applied inappropriately. The reader should carefully monitor the results of tuning the queue manager to be satisfied that there have been no adverse effects.

Customers should test that any changes have not used excessive real resources in their environment and make only essential changes. For example, allocating several megabytes for multiple queues reduces the amount of shared and virtual memory available for other subsystems, as well as over committing real storage.

Note: The 'TuningParameters' stanza is not documented external interface and may change or be removed in future releases.

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. With the queue and log disks configured in this manner, careful consideration must still be given to message persistence: persistent messages should only be used if the message needs to survive a queue manager restart (forced by the administrator or as the result of a power failure, communications failure, or hardware failure). In guaranteeing the recoverability of persistent messages, the pathlength through the queue manager is three times longer than for a nonpersistent message. This overhead does not include the additional time for the message to be written to the log, although this can be minimised by using cached disks.

7.1.1.1 Nonpersistent queue buffer

The default nonpersistent queue buffer size is 64K per queue. This can be increased to 1MB using the TuningParameters stanza and the *DefaultQBufferSize* parameter. Increasing the queue buffer provides the capability to absorb peaks in message throughput at the expense of real storage, but it is **not** suitable as a long term storage for nonpersistent messages as this buffer is **not** recovered after a queue manager restart. Defining queues using large nonpersistent queue buffers can degrade performance if the system is short of real memory either because a large number of queues have already been defined with large buffers, or for other reasons -- e.g. large number of channels defined.

Note: The nonpersistent queue buffer is allocated in shared storage so consideration must be given to whether the agent process or application process has the memory addressability for all the required shared memory segments.

Queues can be defined with different values of *DefaultQBufferSize* and *DefaultQFileSize*. If some queues need to defined differently to others the values can be set in the TuningParameters stanza. 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 from when the queue was created.

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

To improve persistent message throughput the *LogBufferPages* should be increased to its maximum configurable size of *512* x 4K pages = 2MB, the *LogFilePages* (i.e. `crtmqm –lf <LogFilePages>`) should be configured to a large size, for example: *16384* x 4K pages = 64MB, and the number of *LogPrimaryFiles* (i.e. `crtmqm -lp <LogPrimaryFiles>`) should be configured to a large number. The cumulative effect of this tuning will:

- improve the throughput of persistent messages (permitting a possible maximum 2MB of log records to be written from the log buffer to the log disk in a single write).
- reduce the frequency of log switching (permitting a greater amount of log data to be written into one extent).
- allow more time to prepare new linear logs or recycle old circular logs (especially important for long-running units of work).

Changes to the queue manager LogBufferPages stanza take effect at the next queue manager restart. The number of pages can be changed for all subsequent queue managers by changing the LogBufferPages parameter in the product default Log stanza.

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

7.1.3 Channels: process or *thread*, standard or *fastpath*?

It is no longer necessary to consider the system design when deciding whether it is preferable to configure inetd to use process channels (‘amqcrsta’, and for server channels an MCATYPE of ‘PROCESS’), or use threaded channels (‘runmqlsr’, and for server channels an MCATYPE of ‘THREAD’) where prior to Version 5.3 it was necessary to use more than one threaded listener—the *threaded* listener ‘runmqlsr’ **can** now **be used** in **all** scenarios with client and server channels. Additional resource savings are available using the ‘runmqlsr’ listener, including a reduced requirement on: virtual memory, number of processes, file handles, and System V IPC.

Fastpath channels, and/or fastpath applications—see later paragraph 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 performance writing to the log disk: the reader should note that since the greater proportion of time for persistent messages is in the queue manager writing to the log disk, the performance improvement for fastpath channels is less apparent with persistent messages than with nonpersistent messages.

7.2 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 it is recommended that customers use fastpath channels, it is 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 have the capability to run *multiple instances* or *multiple threads* of execution. Although the capacity of a multi-processor (SMP) 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 managers takes to write a group of persistent messages to the log disk. The performance profile of a workload will also be subject to variability through cycles of low and heavy message volumes, 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 for processing nonpersistent 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'), then nonpersistent messages outside of syncpoint do not need to ever be physically placed on a queue.

The reader should note that as more applications are processing messages on a single queue there is an increasing likelihood that queue avoidance will not be maintainable. The reasons for this have a cumulative and exponential effect, for example, 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 being received 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 being retrieved from the queue buffer. While these problems can be addressed by configuring for more MQGETers (i.e processing threads in the server application), or using a larger nonpersistent queue buffer, it may not be possible to avoid a performance degradation.

Processing messages inside syncpoint (i.e. in batches) can be more efficient than outside of syncpoint. As the number of messages in the batch increases, the average processing cost of each message decreases. For persistent messages the queue manager can write the entire batch of messages to the log disk in one go—outside of syncpoint control, the queue manager must wait for each message to be written to the log before returning control to the application.

A typical triggered application follows the performance profile of a short session (refer to '**Short sessions**'—**Page 26**). The 'runmqlsr' has a much smaller overhead of connecting to and disconnecting from the queue manager because it does not have to create a new process. Furthermore, in Version 5.3 the maximum number of connections into a single 'runmqlsr' listener has been significantly increased making it the preferred method of running short sessions over client channels. Although short session performance The programmatical implementation of triggering is still worth consideration with regard to programming a disconnect interval as an input parameter to the application program. This can provide the flexibitity to make tuning adjustments in a production environment, if for instance, it is more efficient to remain connected to the queue manager between periods of message processing, or disconnect to free queue manager and Operating System resources.

7.3 Tuning the Operating System (AIX v4.3.3)

7.3.1 Extended shared memory model: EXTSHM

AIX by default has a slightly different shared memory model to other Unix platforms. This will limit an individual process to 10 shared memory segments being utilised simultaneously.

WebSphere MQ V5.3 makes use of the AIX extension EXTSHM to allow more than 10 segments to be attached to a single process. This is enabled by exporting the **EXTSHM=ON** environment variable before a process is started.

Note: This environment variable must be in upper case.

7.3.2 Scheduling policy: AIXTHREAD_SCOPE

Previous to AIX v4.3.1 XPG5 based applications ran in *system based scheduling* mode. In later AIX versions the application will now run in *process based scheduling* mode. To enable the *system base scheduling* set the environment variable **AIXTHREADS_SCOPE=S** in the */etc/environment* file

Note: This environment variable must be in upper case.

8 Measurement environment

8.1 Workload description

8.1.1 MQI response time tool

The MQI tool exercises the local queue manager by measuring elapsed times of the 8 main MQSeries verbs: MQCONN(X), MQDISC, MQOPEN, MQCLOSE, MQPUT, MQGET, MQCMIT, and MQBACK. The following MQI calls are paired together inside a test application:

- MQCONN(X) with MQDISC
- MQOPEN with MQCLOSE
- MQPUT with MQGET
- MQCMIT and MQBACK with MQPUT and MQGET

Note: MQCLOSE elapsed time is only measured for an empty queue.

Note: Performance of MQCMIT and MQBACK is measured in conjunction with MQPUT and MQGET, putting and getting messages inside a unit of work (i.e. inside syncpoint control). The unit of work is committed at the end of each batch. The number of messages per batch is a parameter of the test.

Note: This tool is not used to measure the performance of verbs: MQSET, MQINQ, or MQBEGIN.

8.1.2 Test scenario workload

8.1.2.1 The driving application programs

The test scenario workload simulates many driving applications running on a single driving machine. This is not typical of a customer environment and is only used to facilitate test coordination. Driving applications were multi-threaded with each thread performing a sequence of MQI calls. The number of threads in each application was adjusted according to whether the test was measuring a local queue manager, a client channel, or distributed queuing scenario. This was done to reduce storage overheads on the driving system. Each driving application thread performed the sequence of actions as outlined in the test scenario illustrations in the '**Performance headlines'** starting on **page 5**.

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, and in the *capacity limit* tests a message rate of 1 round trip per channel per *minute* was used.

Nonpersistent and persistent messages were used in all but the *capacity limit* tests.

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

8.1.2.2 The server application program

The server application is written as a multi-threaded program configured to use 5 threads for processing nonpersistent messages, and 20 or more threads to process persistent messages. Each server thread performed the sequence of actions as outlined in the test scenario illustrations in the '**Performance headlines'** starting on **page 5**.

Nonpersistent messaging is done outside of syncpoint control. Persistent messaging is 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.

8.2 Hardware

IBM M80:	Server system (device under test) / Driving applications machine
Model:	7026-M80
Processor:	500Mhz RS64 III
Architecture:	4-way SMP
	IBM SSA 160 SerialRAID Adapter
Memory (RAM):	4GB
Disk:	2 Internal 16bit SCSI (9GB each, 1 O/S, swap)
	6 SSA Logical disks
	(6 Physical SSA160, 9.1GB, 1 swap, 1 queue, 1 log)
Network:	1GBit Ethernet

8.3 Software

AIX O/S:	AIX v4.3.3
MQSeries:	Version 5.3, and Version 5.2 (Note: queue manager CCSID 819)
Compiler:	C for AIX Compiler, Version 5 (5.0.1.0)

9 Glossary

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* ***runmqlsr*** *=> channels using the 'runmqlsr' listener (client channel test scenario, in addition to 'runmqchi' for distributed queuing test scenarios)* ***c6000*** *=> 6,000 client driving applications (i.e. 6,000 MQI-client connections)* ***q1000*** *=> 1,000 server channel pairs* ***max*** *=> maximum number of channels (or channel pairs)* ***no_correl_id*** *=> correlation identifier not used in the response messages (as each response is placed on a unique reply-to queue per driving application)*
Apps	The number of driving applications connected to the queue manager at the point where the performance measurement is given.
Rate/App/hr	The target message throughput rate of each driving application.
Round T/s	The average achieved message throughput rate of all the driving applications together, measured by the server application.
% (Round T/s)	The percentage increase in the total message throughput rate. *Note: The nature of the comparison is noted under each table where percentage improvements have been given.*
Resp time (s)	The average response time each round trip, as measured and averaged by all the driving applications.
CURDEPTH	The number of messages on the input queue as a snapshot. *Note: runmqsc <qmname>, DISPLAY QLOCAL(<qname>) CURDEPTH*
queue disk (kbps)	The queue disk kilobytes transferred per second.
Swap	The total amount of swap area reservation for all processes in MB, unless otherwise specified as swap/app (i.e. swap area reservation per driving application).
shm	The amount of allocated shared memory in MB.

***** end of document *****