WebSphere MQ for Solaris V6.0 - Performance Evaluations

Version 1.3

August 2005

Peter Toghill .

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

Please take Note!

Before using this report, please be sure to read the paragraphs on “disclaimers”, “warranty and liability exclusion”, “errors and omissions”, and the other general information paragraphs in the "Notices" section below.

First Edition, August 2005.

This edition applies to *WebSphere MQ V6 for Solaris* (and to all subsequent releases and modifications until otherwise indicated in new editions).

Notices

INTENDED AUDIENCE
This report is intended for architects, systems programmers, analysts and programmers

Preface

Target audience

This SupportPac is designed for people who:

- Will be designing and implementing solutions using WebSphere MQ for Solaris.
- Want to understand the performance limits of WebSphere MQ for Solaris V6.0.
- Want to understand what actions may be taken to tune WebSphere MQ for Solaris.

The reader should have a general awareness of the Solaris 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 VI** 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 Richard Eures for his significant contribution to V1.0 of 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.

Unless otherwise specified, the standard message sized used for all the measurements in this report is 2K (2,048 bytes).

A Solaris (model Z801) 4-way UltraSparc-II 440MHz with 4GB of RAM was used as the device under test for all the measurements in this report.

How this document is arranged

Performance Headlines

Pages: 1-17
Section one 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: 21-42
Section two contains performance measurements for *large messages*. This includes *MQI response times* of 50byte to 2MB messages. It also includes *20K, 200k* and *2M* messages using the same scenarios as for the "*Performance Headlines*".

Application Bindings

Page: 43-48
Section three contains performance measurements for *'trusted, normal, and isolated'* server applications, using the same three scenarios as for the "*Performance Headlines*".

Short Sessions

Page: 49

This section was introduced in Version 1.1.

Section four 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.

Performance and Capacity Limits

Pages: 51

This section was introduced in Version V1.1.

Section five 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: 55-57

Measurement Environment

Pages: 59–60
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: 61
A short glossary of the terms used in the tables throughout this document.

CONTENTS

TABLES

FIGURES

1 Overview

WebSphere MQ V6.0 on Solaris has very similar performance characteristics to the V5.3 product. There are several graphs contained in this report that show specific tests have improved or degraded by up to 15% but the majority of comparable measurements are within 5%.

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 do 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

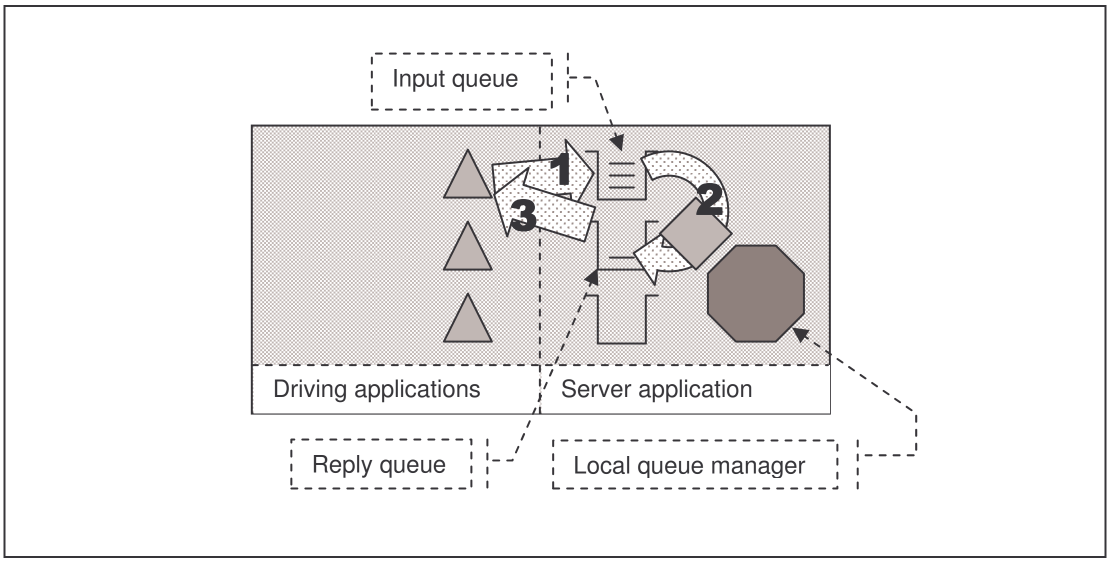


Figure 1 – Connections into a local queue manager

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 the "***Large Messages***" section.

2.1.1 Nonpersistent Messages – Local Queue Manager

Figure 2 and **Figure 3** shows the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario (see **Figure 1** on the previous page), and WebSphere MQ V6.0 compared to Version 5.3.

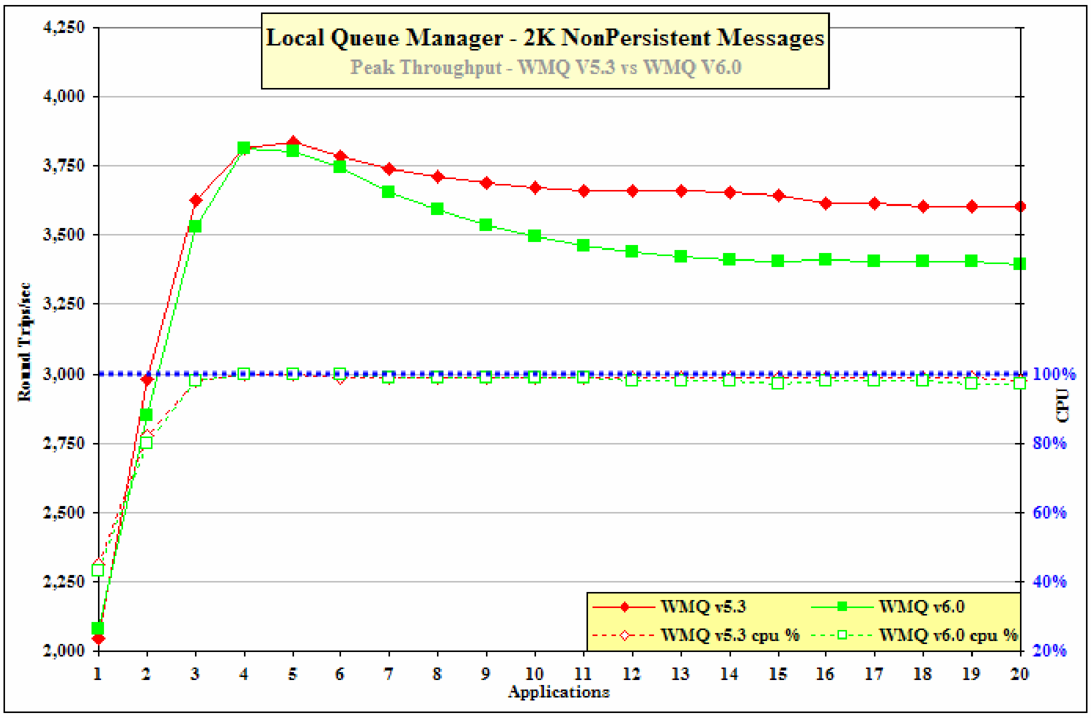


Figure 2 – Performance headline, nonpersistent messages, local queue manager

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

Figure 2 and **Table 1** shows that the peak throughput of nonpersistent messages is similar between Version 5.3 and Version 6.0 (3,837 RT/s – 3,816 RT/s).

Test name: **local_np**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	(4) **5** (20)	(3,816) **3,837** (3,603)	(0.0014) **0.0017** (0.0074)	**100%** (100%) (98%)
WebSphere MQ V6.0	**4** (5) (20)	**3,816** (3802) (3396)	**0.0013** (0.0017) (0.0079)	**100%** (100%) (97%)

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

2.1.2 Persistent Messages – Local Queue Manager

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

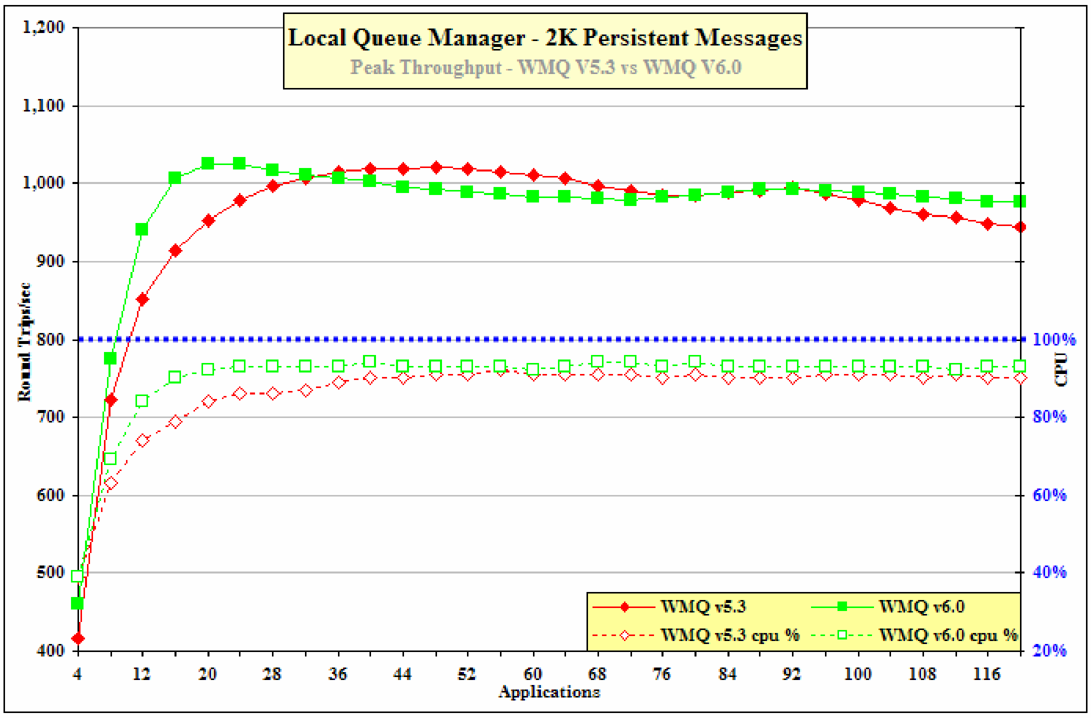


Figure 3 – Performance headline, persistent messages, local queue manager

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

Figure 3 and **Table 2** show that the peak throughput of persistent messages is similar (1,021 RT/s. - 1,025 RT/s) comparing Version 5.3 to Version 6.0. Version 6.0 peaks with fewer applications (20 apps) compared to the number of applications required to peak with Version 5.3 (48 apps).

Test name: **local_pm**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(20) **48** (120)	(952) **1021** (944)	(0.0252) **0.0577** (0.1502)	(84%) **91%** (90%)
WebSphere MQ V6.0	**20** (48) (120)	**1025** (952) (977)	**0.0233** (0.0578) (0.1481)	**92%** (93%) (93%)

Table 2 – Performance headline, persistent 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

2.2 Client Channels Test Scenario

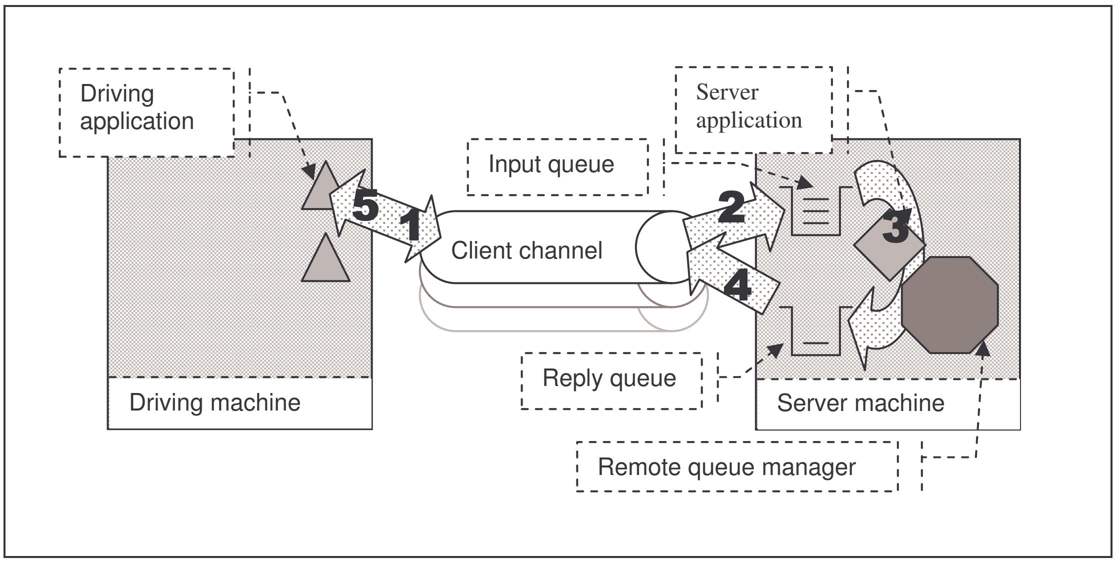


Figure 4 – MQI-client channels into a remote queue manager

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 the "***Large Messages***" section.

2.2.1 Nonpersistent Messages – Client Channels

Figure 5 and **Figure 6** shows the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client channel scenario (see **Figure 4** on the previous page), and WebSphere MQ V6.0 compared to Version 5.3.

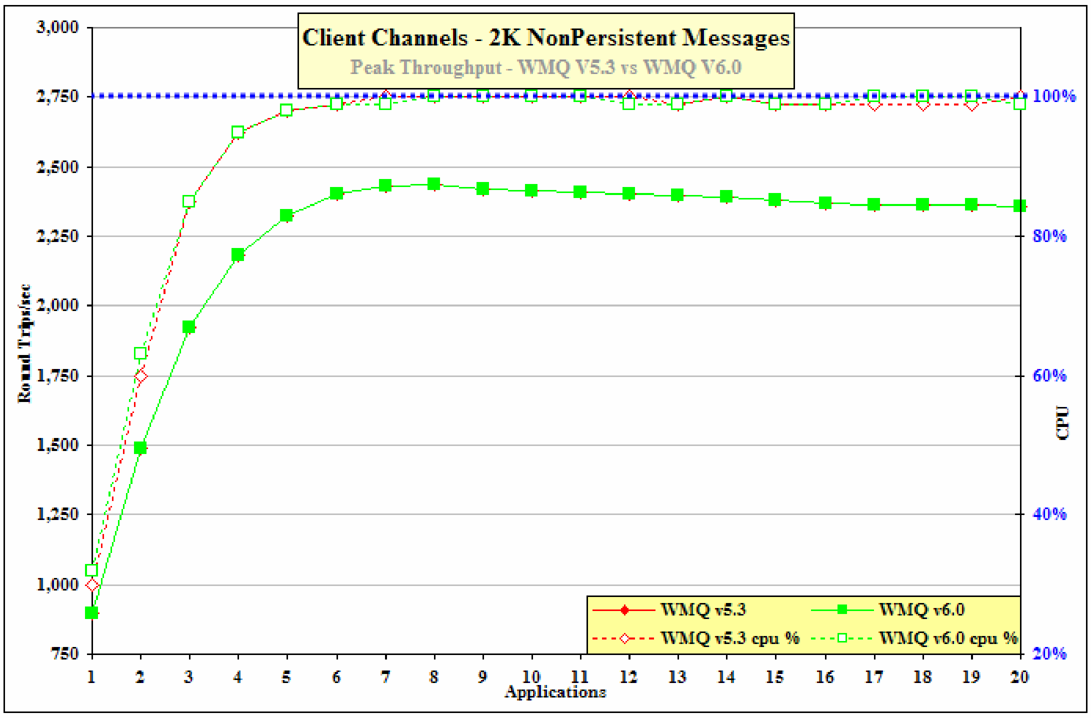


Figure 5 – Performance headline, nonpersistent messages, client channels

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

Figure 5 and **Table 3** show that the peak throughput of nonpersistent messages is similar (2,436 RT/s – 2,467 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **clnp**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(7) **8** (20)	(2,431) **2,436** (2,355)	(0.0033) **0.0038** (0.0099)	(99%) **100%** (99%)
WebSphere MQ V6.0	**7** (8) (20)	**2,467** (2,464) (2,291)	**0.0033** (0.0038) (0.0102)	**100%** (100%) (100%)

Table 3 – Performance headline, nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.*

2.2.2 Persistent Messages – Client Channels

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

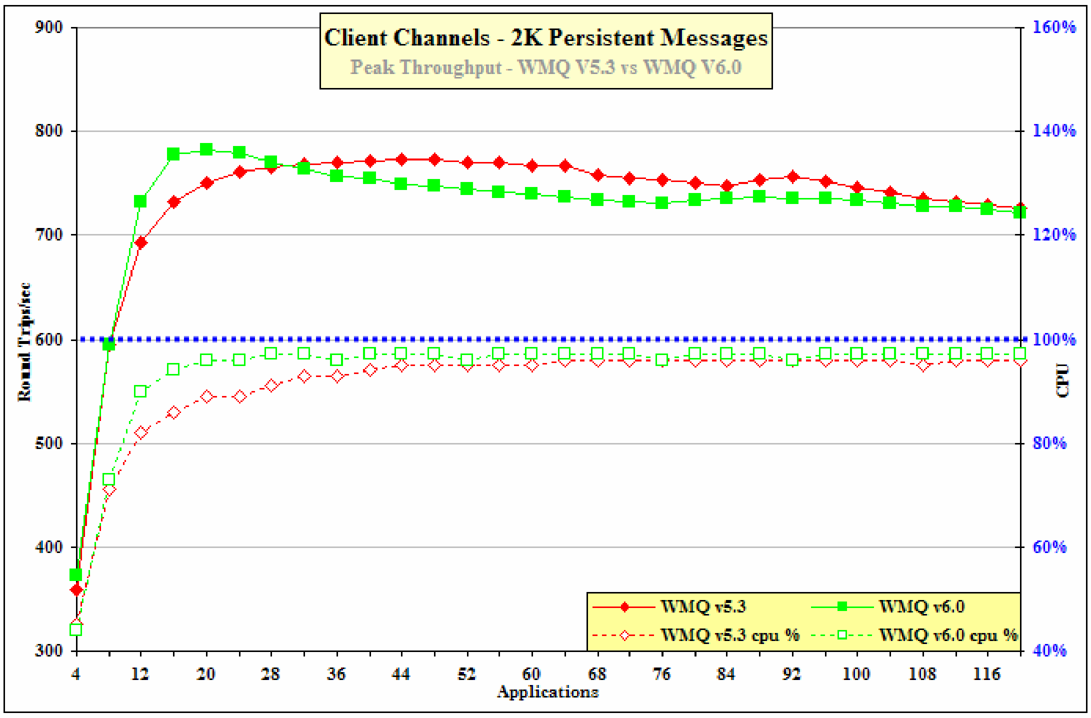


Figure 6 – Performance headline, persistent messages, client channels

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

Figure 6 and **Table 4** show that the peak throughput of persistent messages is similar (773 RT/s - 782 RT/s) comparing Version 5.3 to Version 6.0. Version 6.0 peaks with fewer applications (20 apps) compared to the number of applications required to peak with Version 5.3 (48 apps).

Test name: **clpm**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(20) **48** (120)	(751) **773** (727)	(0.0305) **0.0728** (0.1946)	(89%) **97%** (96%)
WebSphere MQ V6.0	**20** (48) (120)	**782** (748) (722)	**0.0301** (0.0773) (0.2009)	**96%** (97%) (97%)

Table 4 – Performance headline, persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

2.2.3 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.

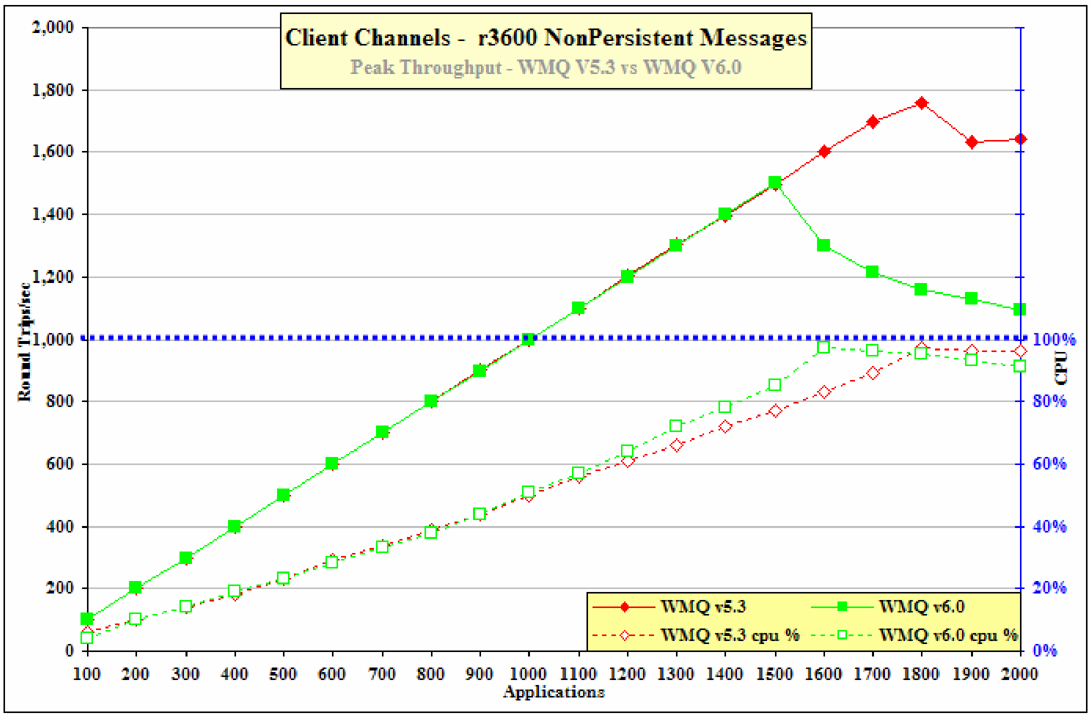


Figure 7 – 1 round trip per driving application per second, client channels, nonpersistent messages

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

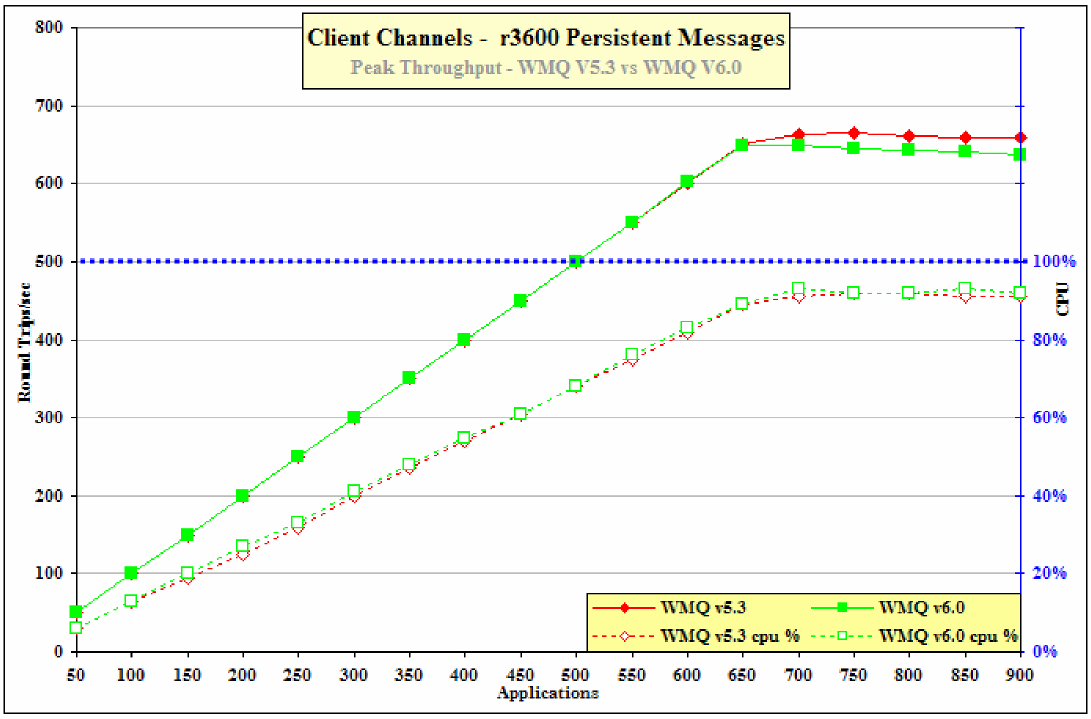


Figure 8 – 1 round trip per driving application per second, client channels, persistent messages

Figure 7, Figure 8 and **Table 5** (next page) shows how WebSphere MQ V6.0 has an 11% reduction in the number of highly active MQI-client connections processing non-persistent messages into a single queue manager whereas persistent messages have similar performance.

Test name:	Apps	Rate/app/hr	Round Trips/sec	Response time (s)	CPU
clnp_r3600 (WebSphereMQ v5.3)	**1,500** (1,700)	3,600	**1,499** (1,699)	**0.3229** (0.1002)	**85%** (89%)
clpm_r3600 (WebSphereMQ v5.3)	**650** (650)	3,600	**649** (650)	**0.2654** (0.5271)	**89%** (89%)

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

Note: The large bold numbers in the table above show the WebSphere MQ V6.0 peak number of round trips per second, and the number of driving applications used to achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison with Version 5.3.

Figure 9 and **Figure 10** shows the increased memory requirement of an MQI-client connection using the runmqlsr listener in WebSphere MQ V6.0.

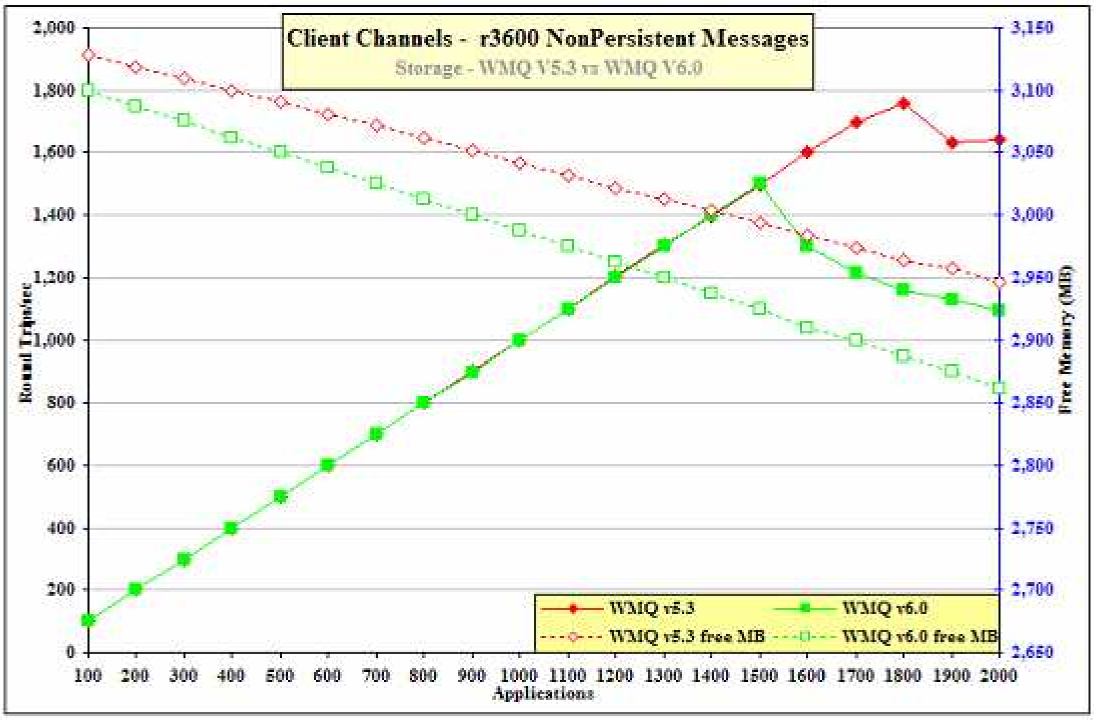


Figure 9 – Free memory, client channels, nonpersistent messages

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

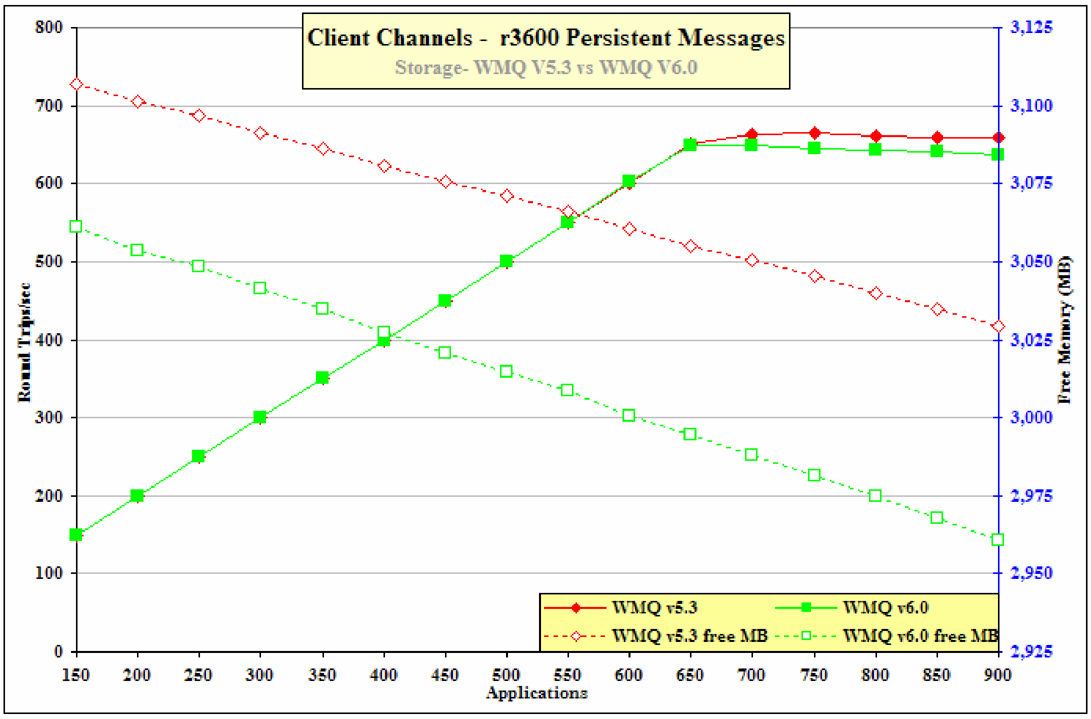


Figure 10 – Free memory, client channels, persistent messages

Test name:	Apps	Free (MB)
clnp_r3600 (WebSphereMQ v5.3)	**100 / 1,500** (100 / 1,500)	**3,100 / 2,924** (3,129 / 2,994)
clpm_r3600 (WebSphereMQ v5.3)	**150 / 650** (150 / 650)	**3,061 / 2,994** (3,107 / 3,055)

Table 6 – Free memory, client channels

Note: The large bold numbers in the table above show the WebSphere MQ V6.0 peak number of round trips per second, and the number of driving applications used to achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison with Version 5.3.

Note: The free memory shown in ***Table 6*** *represents the available real memory, not swap memory.*

The amount of free memory consumed per channel has increased by 30Kbytes. Clients processing non persistent messages need 126K and clients processing persistent messages require 166K. This increase does effect the size of the swap file but the page manager will eliminate those pages that are not referenced send them out to secondary storage.

For further calculations on the swap reservation and shared memory utilisation, refer to '**Performance and Capacity Limits**'. This will be covered in the next version of this document.

2.3 Distributed Queuing Test Scenario

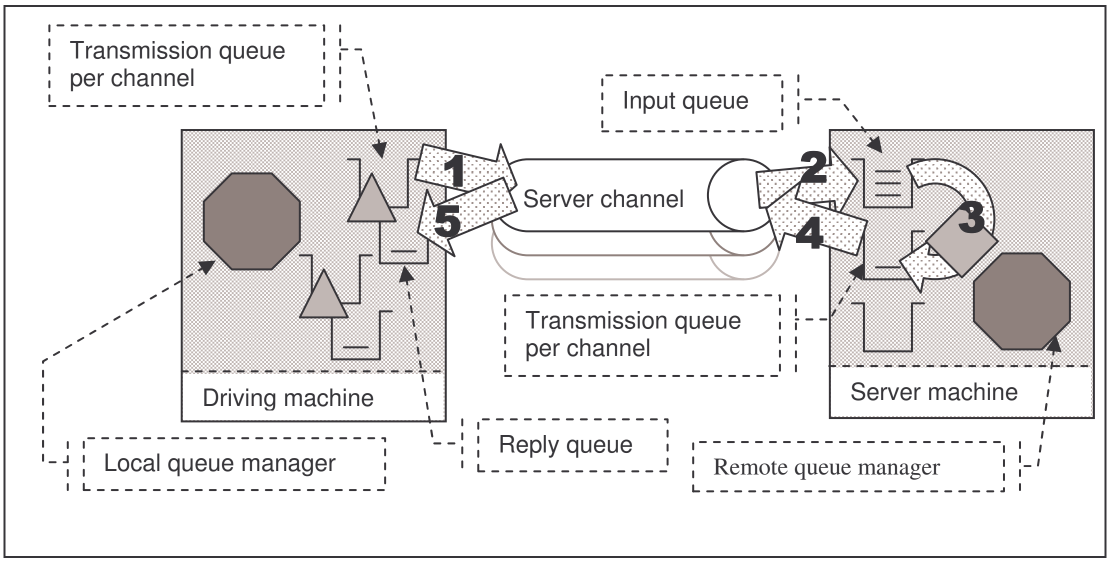


Figure 11 – Server channels between two queue managers

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 the "***Large Messages***" section.

2.3.1 Nonpersistent Messages – Server Channels

Figure 12 and **Figure 13** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario (see **Figure 11** on the previous page), and WebSphere MQ V6.0 compared to Version 5.3.

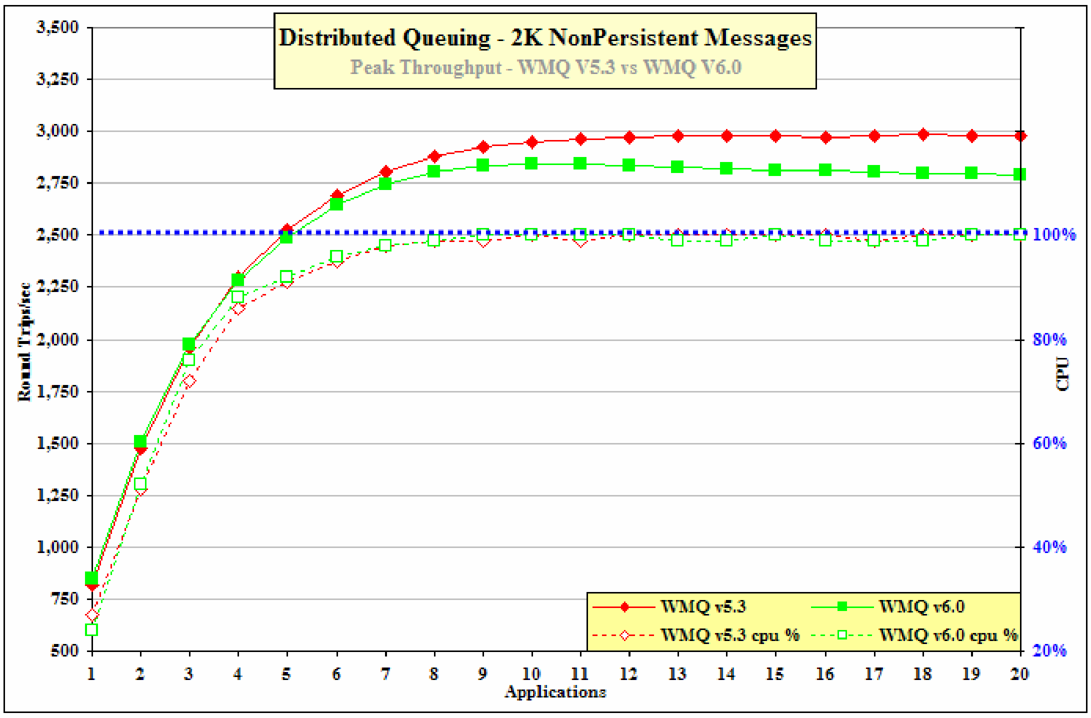


Figure 12 – Performance headline, nonpersistent messages, server channels

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

Figure 12 and **Table 7** show that the peak throughput of nonpersistent messages has decreased by 4.7% (2,984 RT/s – 2,844 RT/s) comparing Version 5.3 to Version 6.0.

Test name: dqnp	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(10) **18** (20)	(2,951) **2,984** (2,978)	(0.0039) **0.0071** (0.0078)	(100%) **100%** (100%)
WebSphere MQ V6.0	**10** (18) (20)	**2,844** (2,796) (2,788)	**0.0041** (0.0077) (0.0085)	**100%** (99%) (100%)

Table 7 – Performance headline, nonpersistent messages, server channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.*

2.3.2 Persistent Messages – Server Channels

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

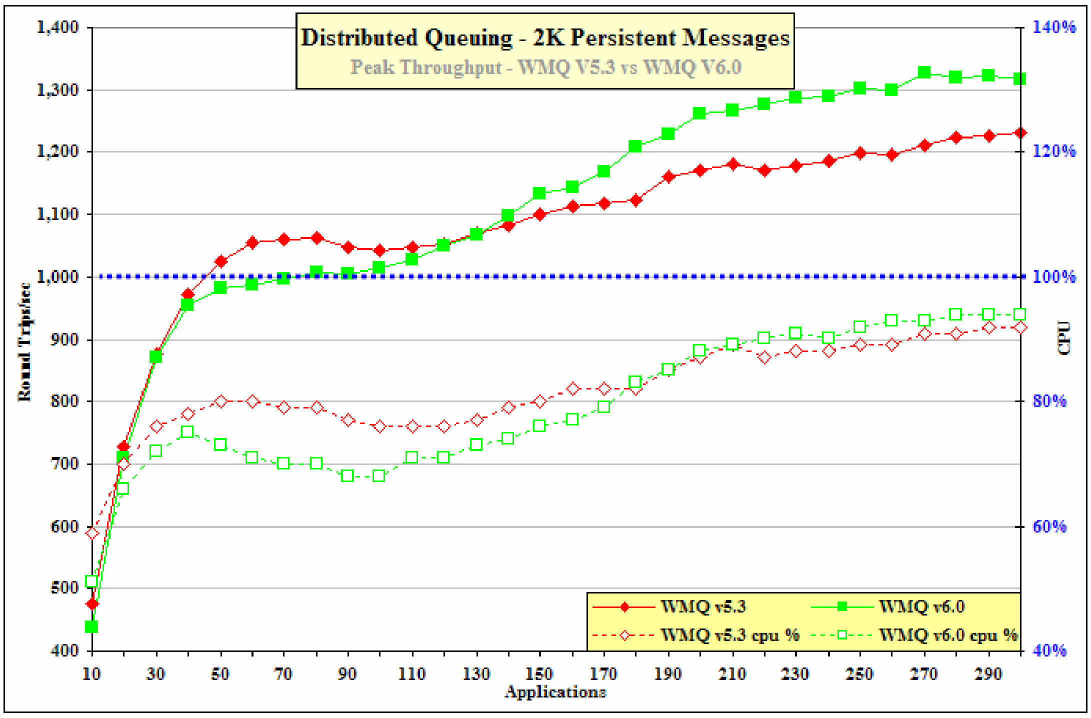


Figure 13 – Performance headline, persistent messages, server channels

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

Figure 13 and **Table 8** show that the peak throughput of persistent messages has increased by 7.8% (1,232 RT/s – 1,328 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **dqpm**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(270) **300**	(1,211) **1232**	(0.2249) **0.2878**	(91%) **92%**
WebSphere MQ V6.0	**270** (300)	**1,328** (1,317)	**0.2223** (0.2588)	**93%** (94%)

Table 8 – Performance headline, persistent messages, server channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

2.3.3 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.

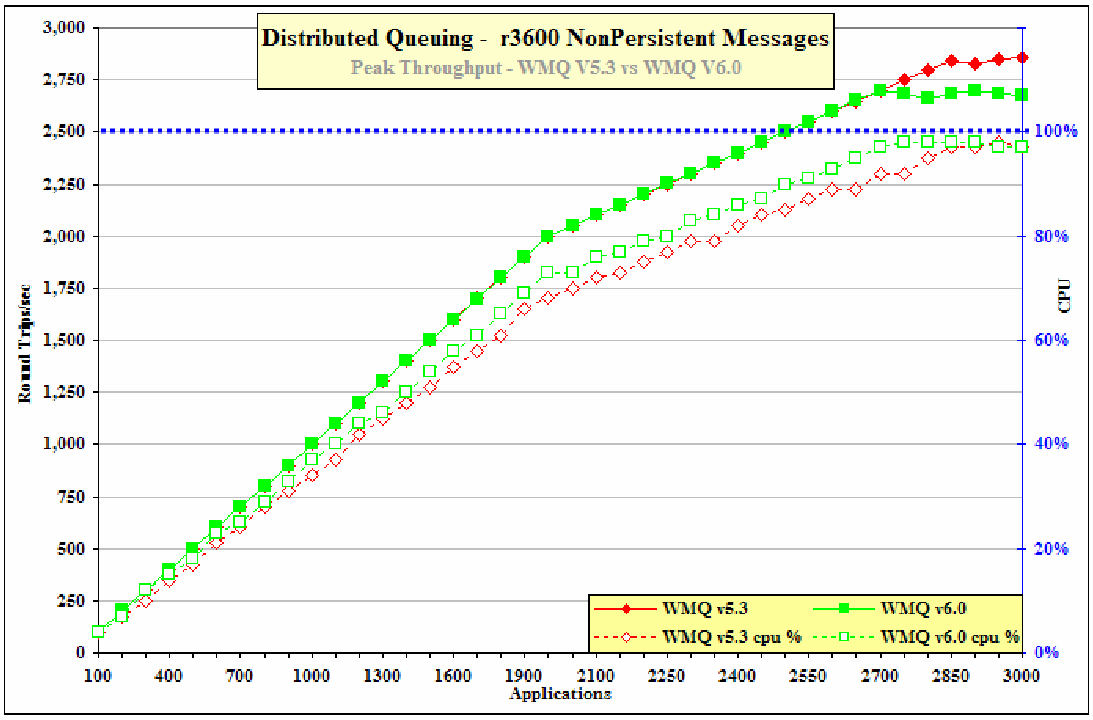


Figure 14 – 1 round trip per driving application per second, server channel, nonpersistent messages

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

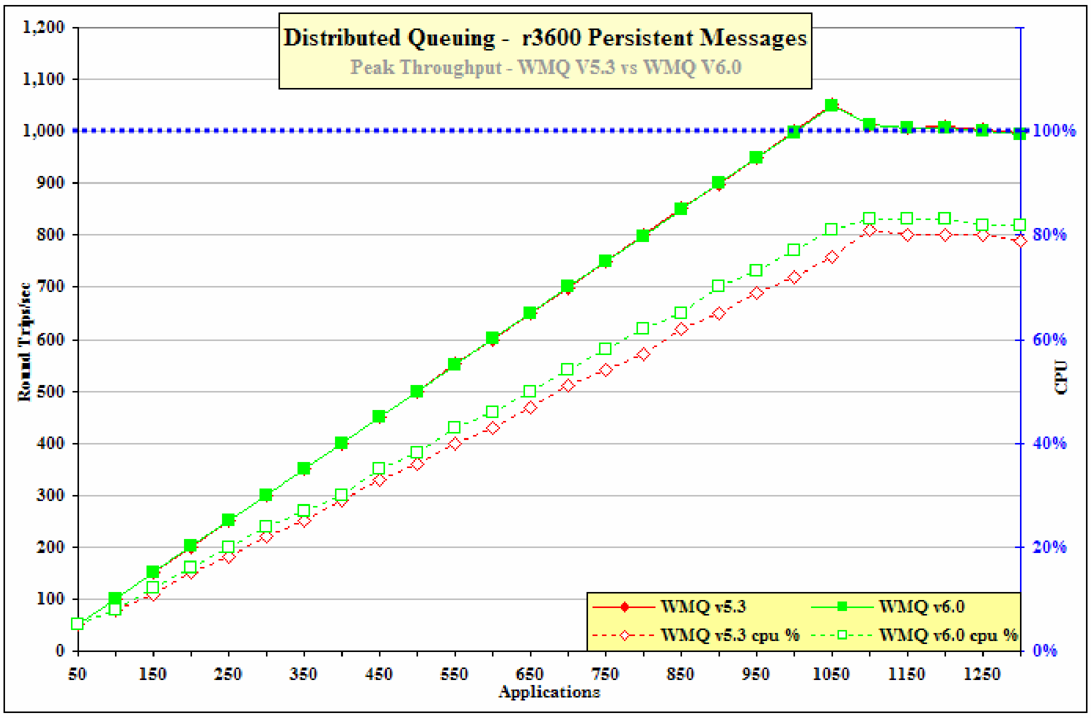


Figure 15 – 1 round trip per driving application per second, server channel, persistent messages

Figure 14, **Figure 15** and **Table 9** (next page) shows how WebSphere MQ V6.0 has an 5.7% reduction in performance as version 5.3 and persistent messages is similar.

Test name:	Apps	Rate/app/hr	Round Trips/sec	Response time (s)	CPU
dqnp_r3600 (WebSphereMQ v5.3)	**2,700** (3,100)	3,600	**2,700** (2,866)	**0.1178** (1.3033)	**97%** (98%)
dqpm_r3600 (WebSphereMQ v5.3)	**1,050** (1,050)	3,600	**1,050** (1,051)	**0.4310** (0.6133)	**81%** (76%)

Table 9 – 1 round trip per driving application per second, client channels

Note: The large bold numbers in the table above show the WebSphere MQ V6.0 peak number of round trips per second, and the number of driving applications used to achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison with Version 5.3.

3 Large Messages

3.1 MQI Response Times: 50bytes to 100MB – Local Queue Manager

Queue manager log configuration:

LogPrimaryFiles=3, LogFilePages=2048

3.1.1 50bytes to 32KB

Figure 16 show that the response time for MQPUT/GET pairs is slightly slower for all nonpersistent message sizes between 50bytes and 32KB.

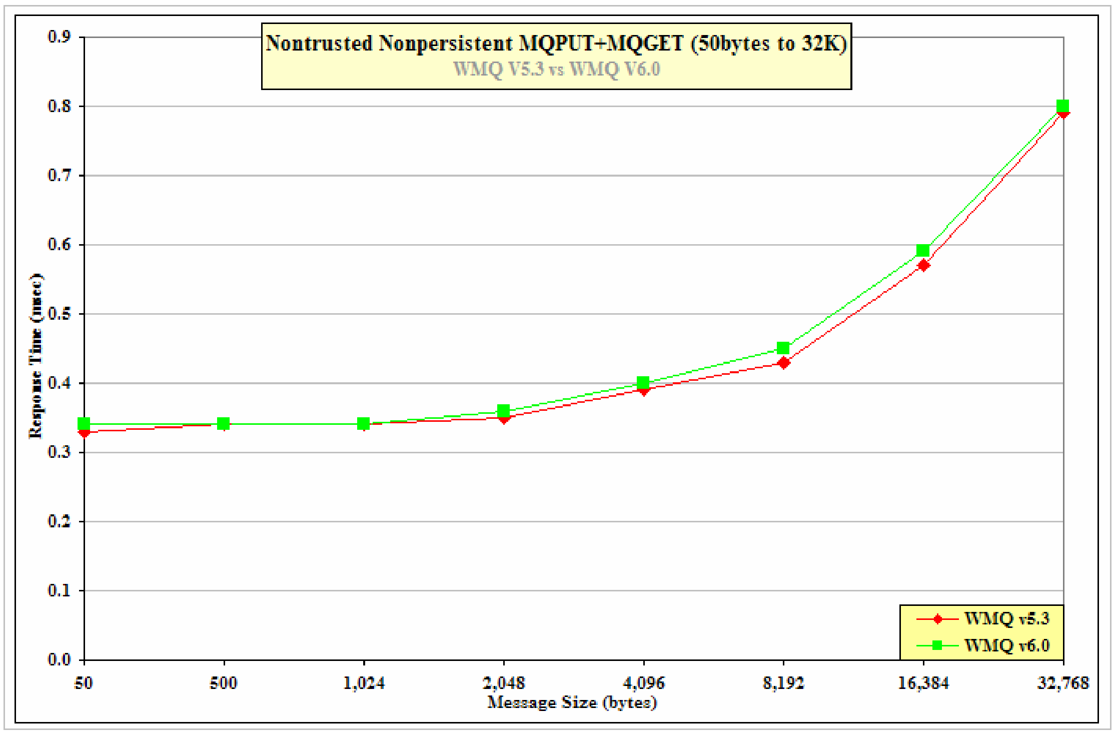


Figure 16 –The effect of nonpersistent message size on MQI response time (50byte - 32K)

Figure 17 show that the response for MQPUT/GET pairs is slightly slower for all persistent message sizes between 50bytes and 16KB.

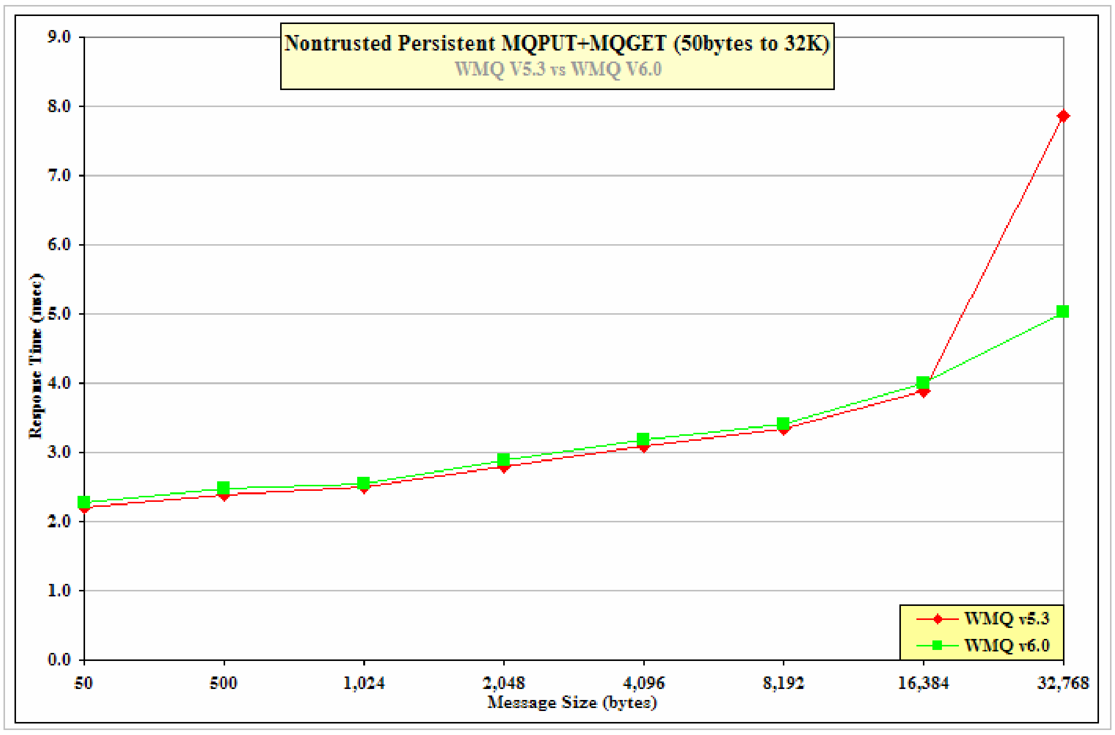


Figure 17 –The effect of persistent message size on MQI response time (50byte - 32K)

3.1.2 32KB to 2MB

Figure 18 show that the response time for MQPUT/GET pairs is slightly quicker for all nonpersistent message sizes between 64KB and 2MB.

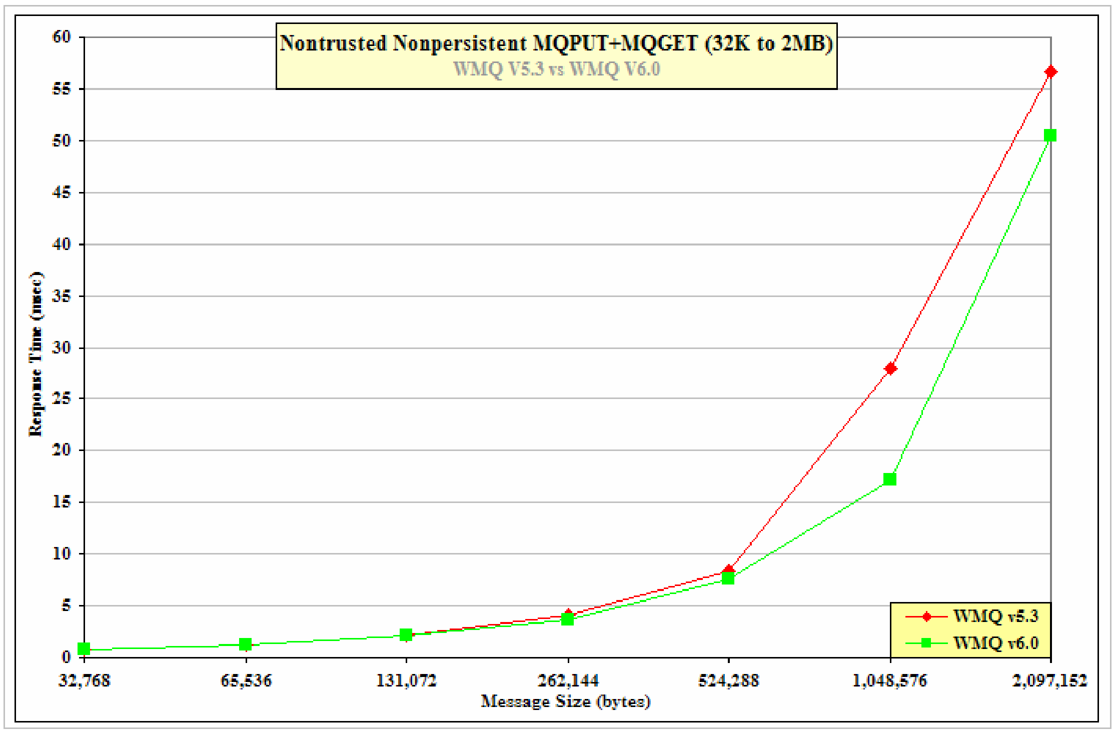


Figure 18 –The effect of nonpersistent message size on MQI response time (32K – 2MB)

Figure 19 show that the response for MQPUT/GET pairs is improved for all persistent message sizes between 32KB and 2MB.

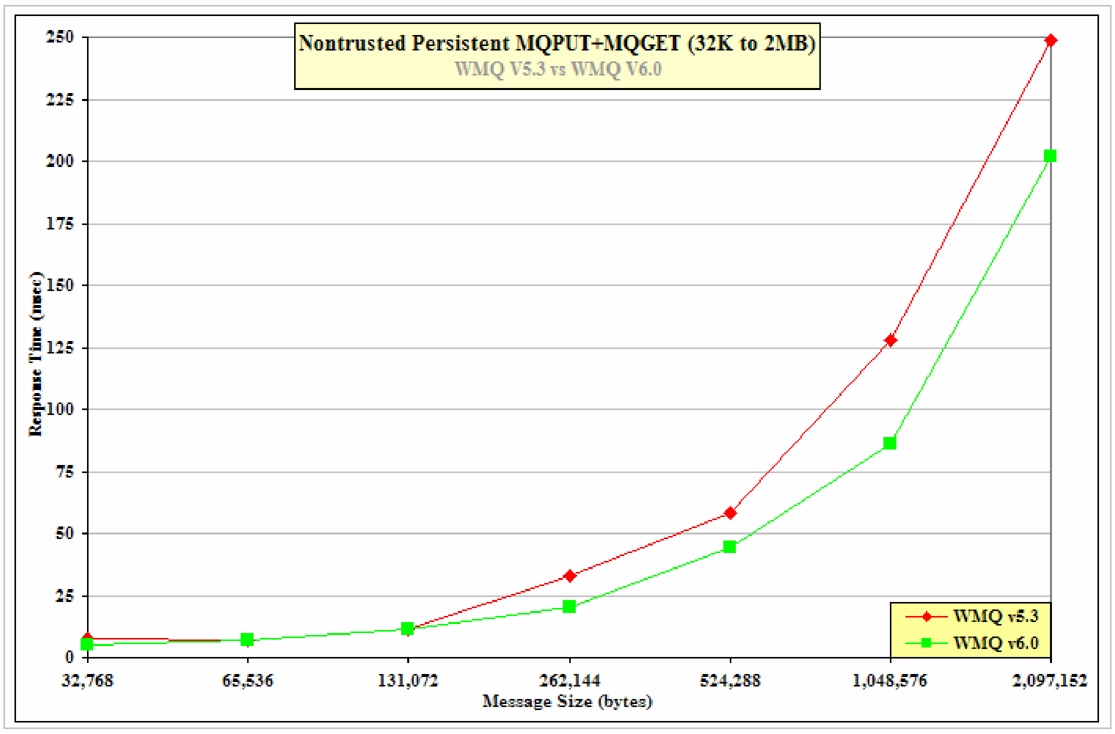


Figure 19 –The effect of persistent message size on MQI response time (32K – 2MB)

3.1.3 2MB to 100MB

Figure 20 show that the response time for MQPUT/GET pairs is slightly quicker for all nonpersistent message sizes between 2MB and 100Mb.

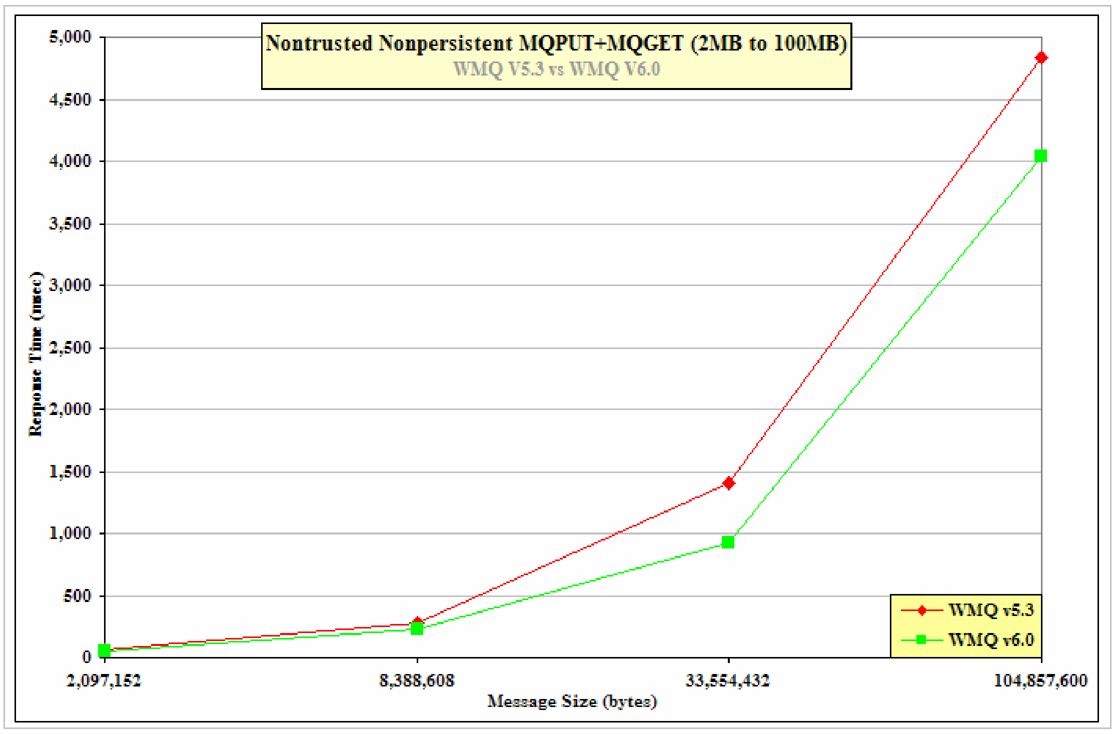


Figure 20 –The effect of nonpersistent message size on MQI response time (2MB – 100MB)

Figure 21 show that the response for MQPUT/GET pairs is improved for all persistent message sizes between 2MB and 100MB.

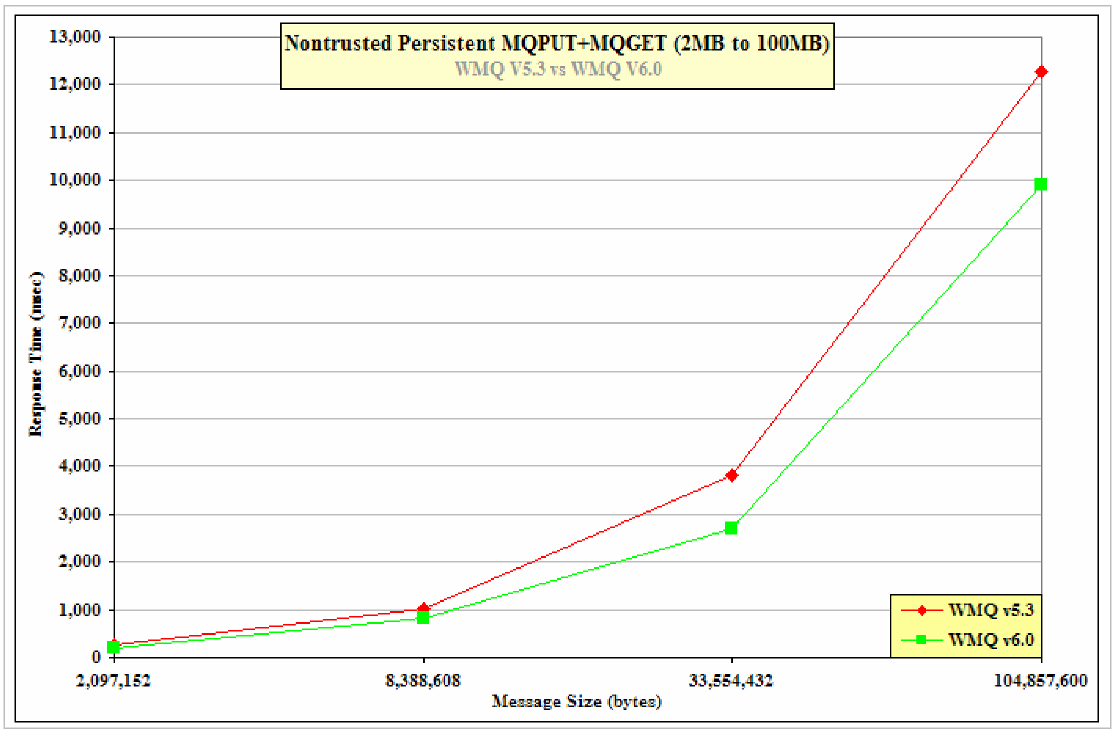


Figure 21 –The effect of persistent message size on MQI response time (2MB – 100MB)

3.2 20K Messages

3.2.1 Local Queue Manager

Figure 22 and **Figure 23** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario.

3.2.1.1 Nonpersistent Messages

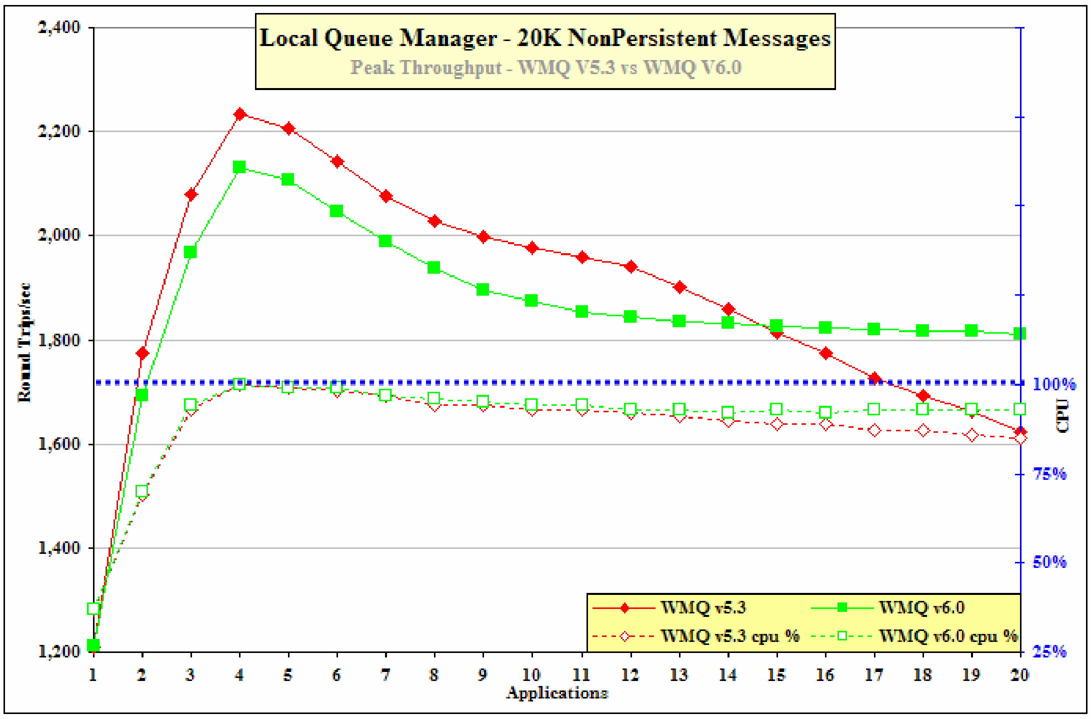


Figure 22 – 20K nonpersistent messages, local queue manager

Figure 22 and **Table 10** show that the peak throughput of nonpersistent messages has decreased by 4.7% (2,235 RT/s – 2,129 RT/s) comparing Version 5.3 to Version 6.0.

Test name: local_np_20K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**4** (20)	**2,235** (1,622)	**0.0022** (0.0151)	**100%** (85%)
WebSphere MQ V6.0	**4** (20)	**2,129** (1,811)	**0.0024** (0.0147)	**100%** (93%)

Table 10 – 20K 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3*

3.2.1.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

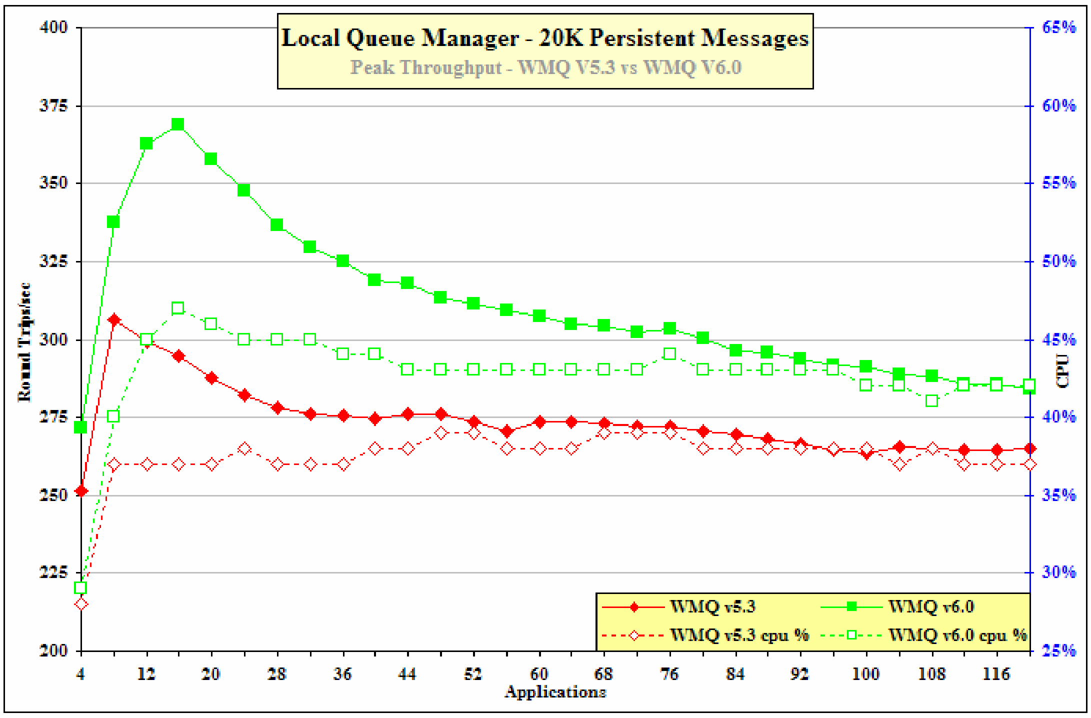


Figure 23 – 20K persistent messages, local queue manager

Figure 23 and **Table 11** show that the peak throughput of persistent messages has increased by 19.4% (306 RT/s – 369 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **local_pm_20K**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	**8** (16) (120)	**306** (295) (265)	**0.0284** (0.0615) (0.5378)	**37%** (37%) (37%
WebSphere MQ V6.0	(8) **16** (120)	(338) **369** (284)	(0.0500) **0.0261** (0.5036)	(40%) **47%** (42%)

Table 11 – 20K persistent 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.2.2 Client Channel

Figure 24 and **Figure 25** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client channel scenario.

3.2.2.1 Nonpersistent Messages

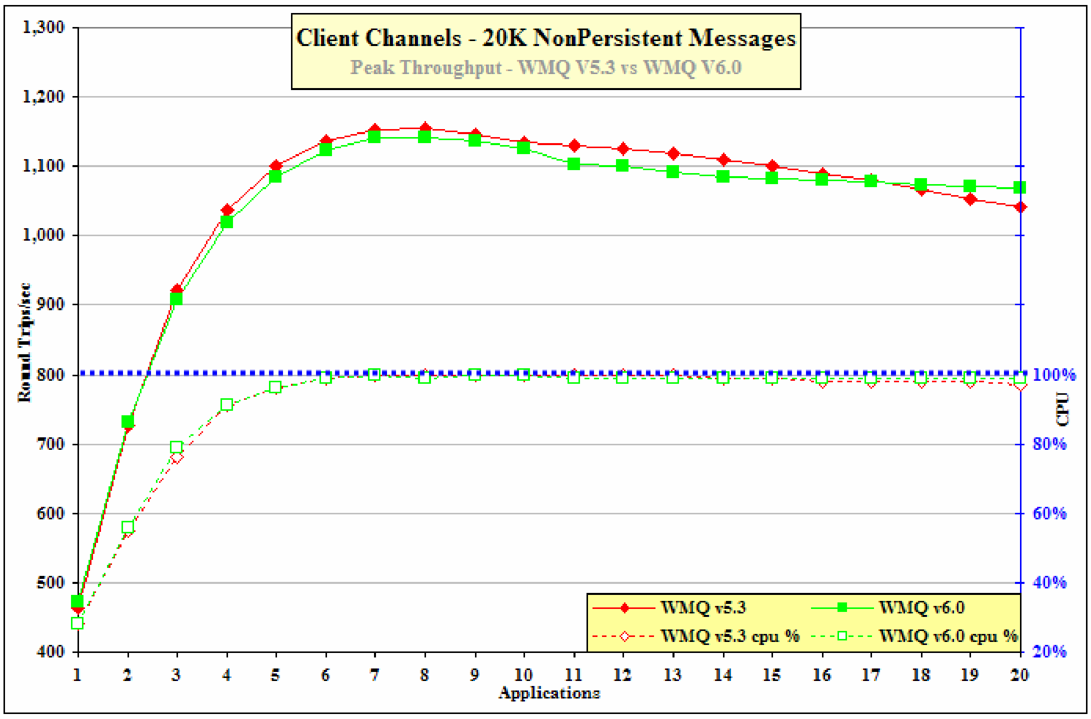


Figure 24 – 20K nonpersistent messages, client channels

Figure 24 and **Table 12** show that the peak throughput of nonpersistent messages is similar (1,154 RT/s – 1,141 RT/s) comparing Version 5.3 to Version 6.0.

Test name: clnp_20K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(7) **8** (20)	(1,152) **1,154** (1,042)	(0.0070) **0.0080** (0.0231)	(100%) **100%** (97%)
WebSphere MQ V6.0	**7** (8) (20)	**1,141** (1,141) (1,068)	**0.0070** (0.0081) (0.0221)	**100%** (99%) (99%)

Table 12 – 20K nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.2.2.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

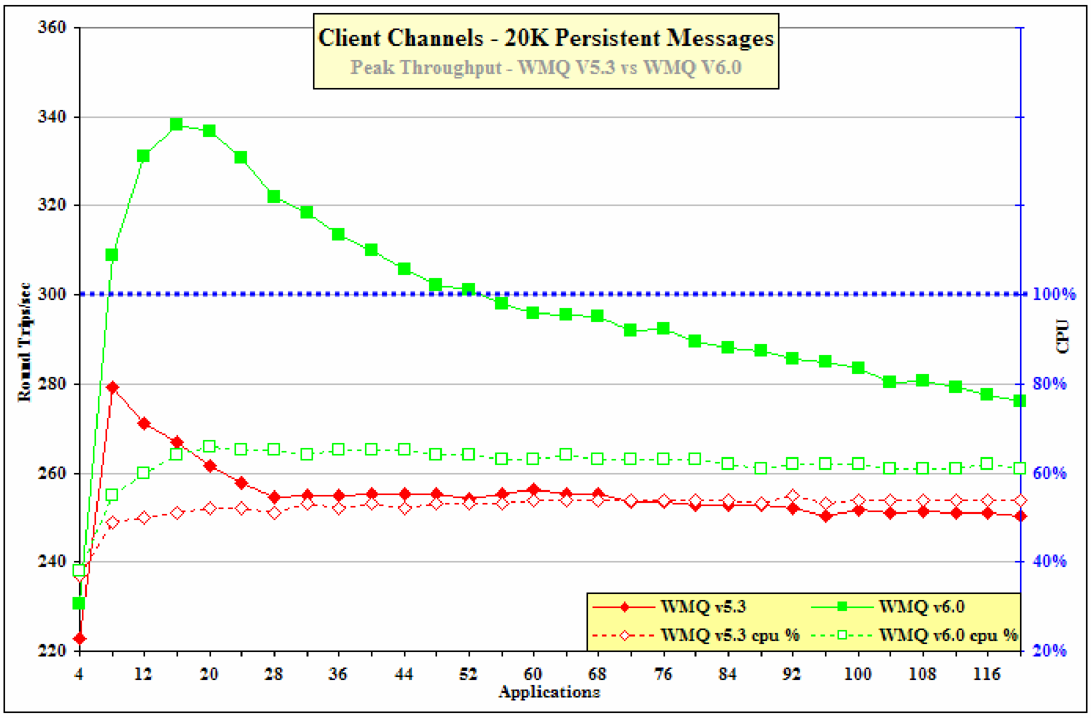


Figure 25 – 20K persistent messages, client channels

Figure 25 and **Table 13** show that the peak throughput of persistent messages has increased by 21.1% (279 RT/s – 338 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **clpm_20K**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	**8** (16) (120)	**279** (267) (250)	**0.0313** (0.0689) (0.5710)	**48%** (51%) (54%)
WebSphere MQ V6.0	(8) **16** (120)	(309) **338** (276)	(0.0288) **0.0541** (0.5151)	(55%) **64%** (61%)

Table 13 – 20K persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.2.3 Distributed Queuing

Figure 26 and **Figure 27** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario

3.2.3.1 Nonpersistent Messages

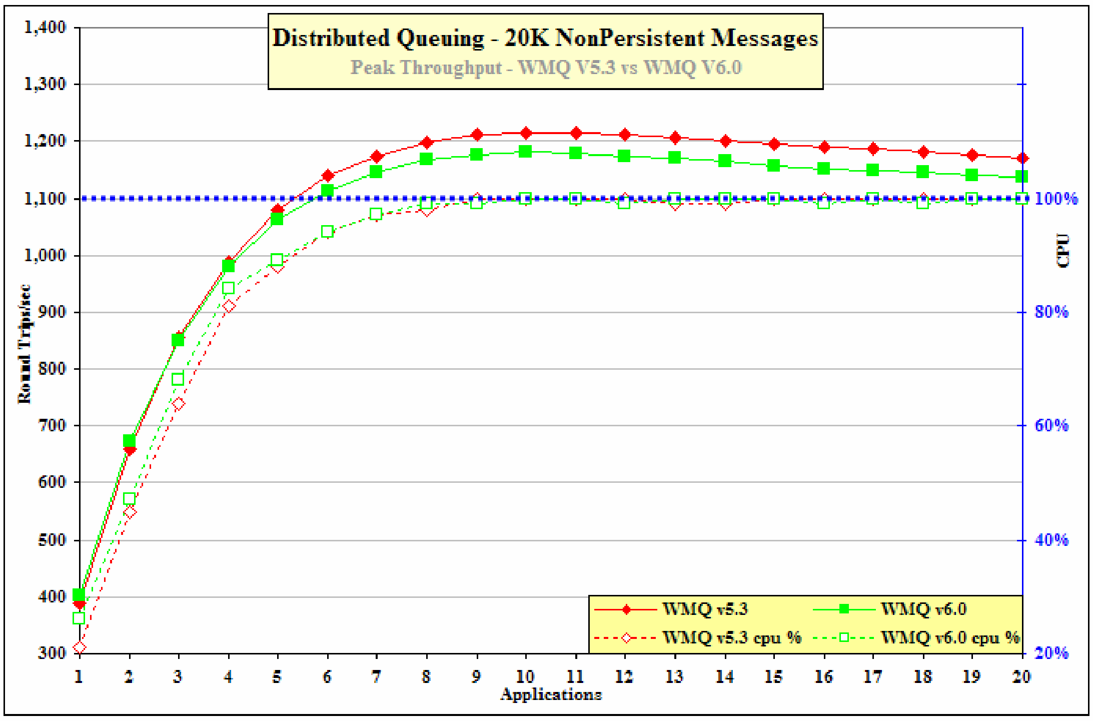


Figure 26 – 20K nonpersistent messages, distributed queuing

Figure 26 and **Table 14** show that the peak throughput of nonpersistent messages has decreased by 2.9% (1,216 RT/s – 1,181 RT/s) comparing Version 5.3 to Version 6.0.

Test name: dqnp_20K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**10** (20)	**1,216** (1,170)	**0.0096** (0.0202)	**100%** (100%)
WebSphere MQ V6.0	**10** (20)	**1,181** (1,138)	**0.0099** (0.0208)	**100%** (100%)

Table 14 – 20K nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.2.3.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

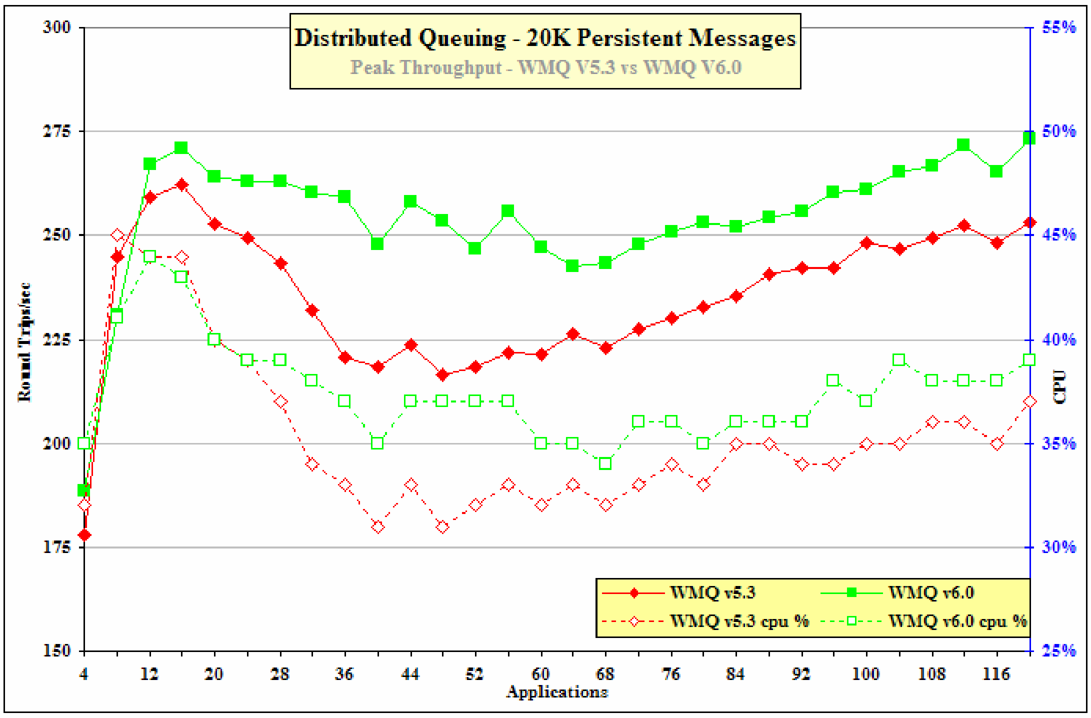


Figure 27 – 20K persistent messages, distributed queuing

Figure 27 and **Table 15** show that the peak throughput of nonpersistent messages has increased by 4.2% (262 RT/s – 273RT/s) comparing Version 5.3 to Version 6.0.

Test name: dqpm_20K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**16** (120)	**262** (253)	**0.0703** (0.5645)	**44%** (37%)
WebSphere MQ V6.0	(16) **120**	(271) **273**	(0.0687) **0.5240**	(43%) **39%**

Table 15 – 20K persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.3 200K Messages

3.3.1 Local Queue Manager

Figure 28 and **Figure 29** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario.

3.3.1.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize =4000000

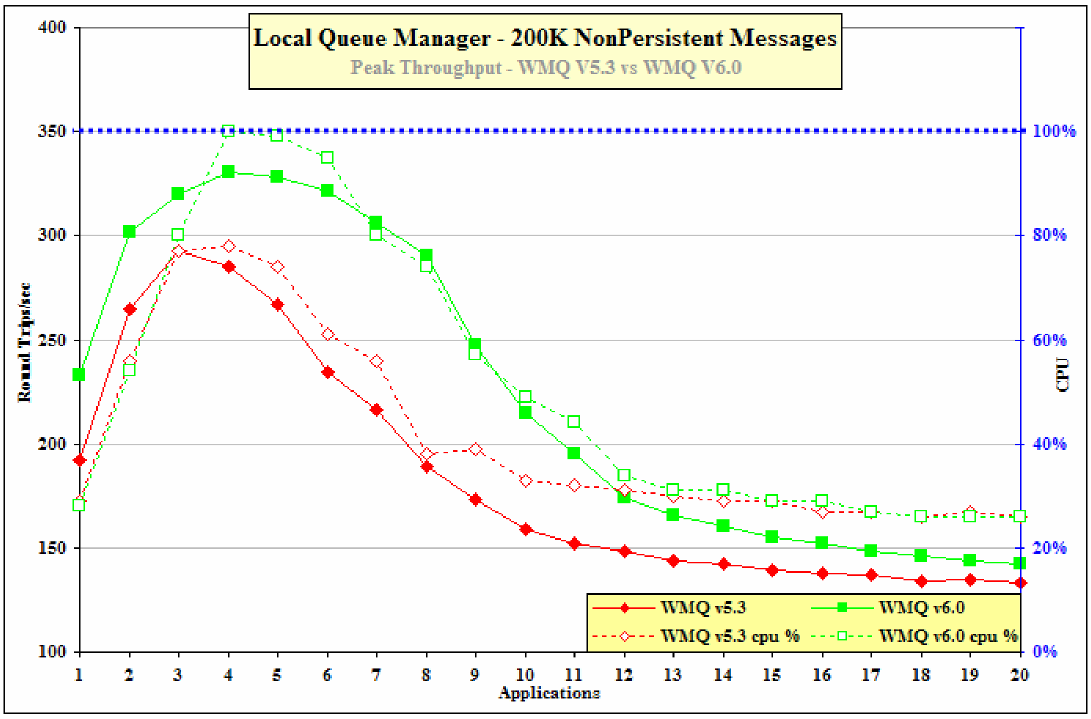


Figure 28 – 200K nonpersistent messages, local queue manager

Figure 28 and **Table 16** show that the peak throughput of nonpersistent messages has increased by 13.0% (293 RT/s – 331 RT/s) comparing Version 5.3 to Version 6.0.

Test name: local_np_200K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**3** (4) (20)	**293** (285) (133)	**0.0116** (0.0162) (0.1791)	**77%** (78%) (26%)
WebSphere MQ V6.0	(3) **4** (20)	(320) **331** (142)	(0.0116) **0.0139** (0.1669)	(80%) **100%** (26%)

Table 16 – 200K 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3

3.3.1.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

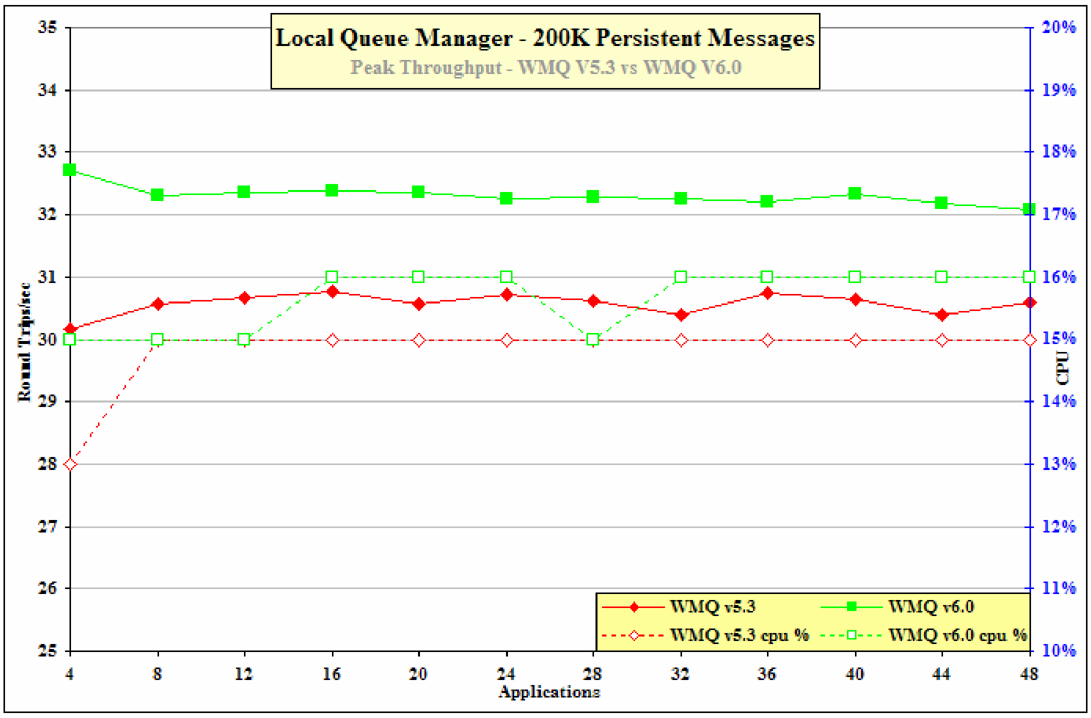


Figure 29 – 200K persistent messages, local queue manager

Figure 29 and **Table 17** show that the peak throughput of persistent messages has increased by 6.3% (31 RT/s – 33 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **local_pm_200K**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(4) **16** (48)	(30) **31** (31)	(0.1322) **0.5890** (1.84656)	(13%) **15%** (15%)
WebSphere MQ V6.0	**4** (16) (48)	**33** (32) (32)	**0.1219** (0.5619) (1.7831)	**15%** (16%) (16%)

Table 17 – 200K persistent 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.3.2 Client Channel

Figure 30 and **Figure 31** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client channel scenario.

3.3.2.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize =4000000

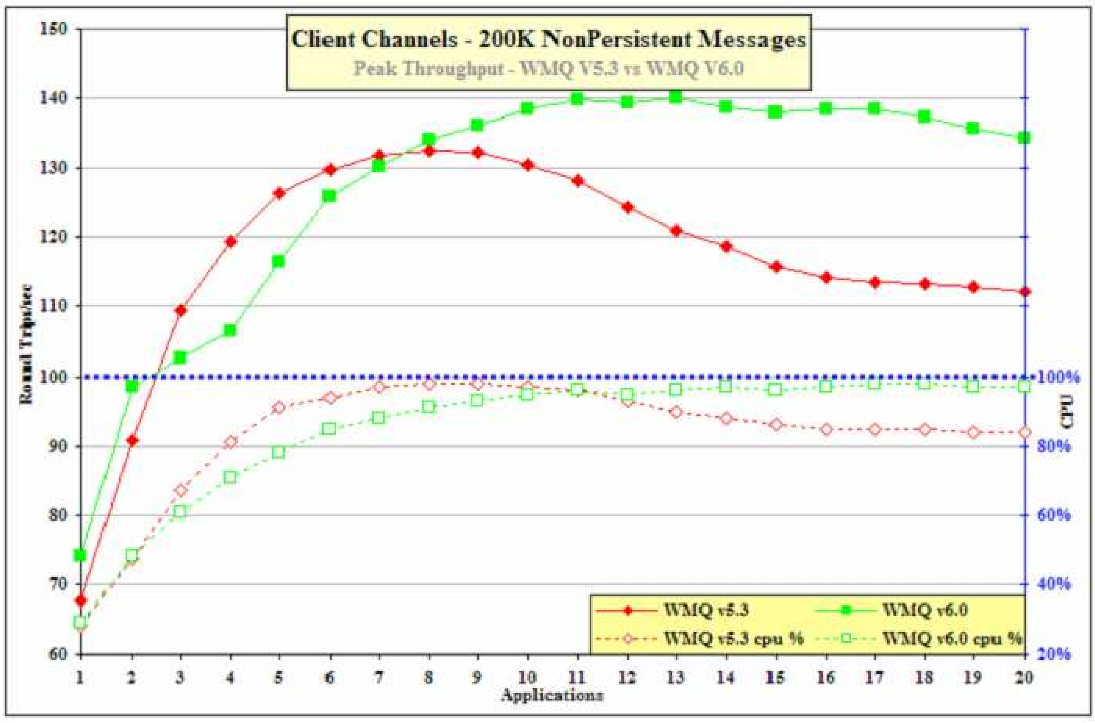


Figure 30 – 200K nonpersistent messages, client channels

Figure 30 and **Table 18** show that the peak throughput of nonpersistent messages has increased by 5.7% (133 RT/s – 140 RT/s) comparing Version 5.3 to Version 6.0.

Test name: clnp_200K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**8** (13) (20)	**133** (121) (112)	**0.0703** (0.1259) (0.2095)	**98%** (90%) (84%)
WebSphere MQ V6.0	(8) **13** (20)	(134) **140** (134)	(0.0696) **0.1096** (0.1713)	(91%) **96%** (97%)

Table 18 – 200K nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.3.2.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

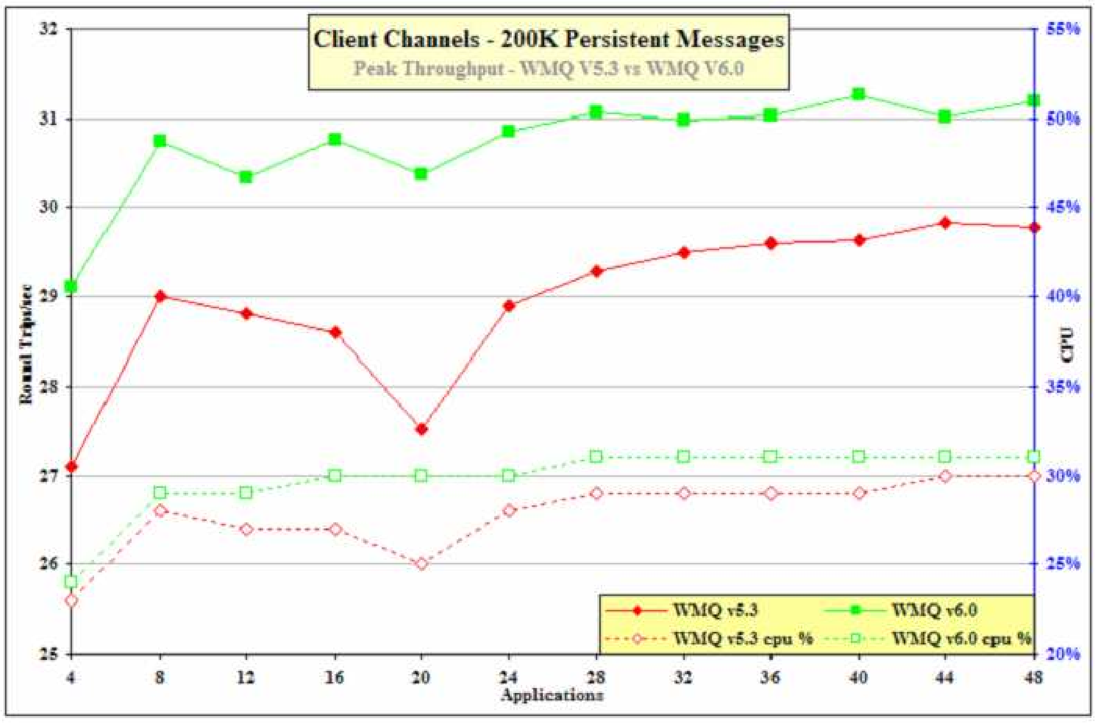


Figure 31 – 200K persistent messages, client channels

Figure 31 and **Table 19** show that the peak throughput of persistent messages has increased by 4.8% (30 RT/s – 31 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **clpm_200K**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	(40) **44** (48)	(30) **30** (30)	(1.5748) **1.7352** (1.8770)	(29%) **30%** (30%)
WebSphere MQ V6.0	**40** (44) (48)	**31** (31) (31)	**1.5196** (1.6666) (1.8039)	**31%** (31%) (31%)

Table 19 – 200K persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.*

3.3.3 Distributed Queuing

Figure 32 and **Figure 33** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario

3.3.3.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize =4000000

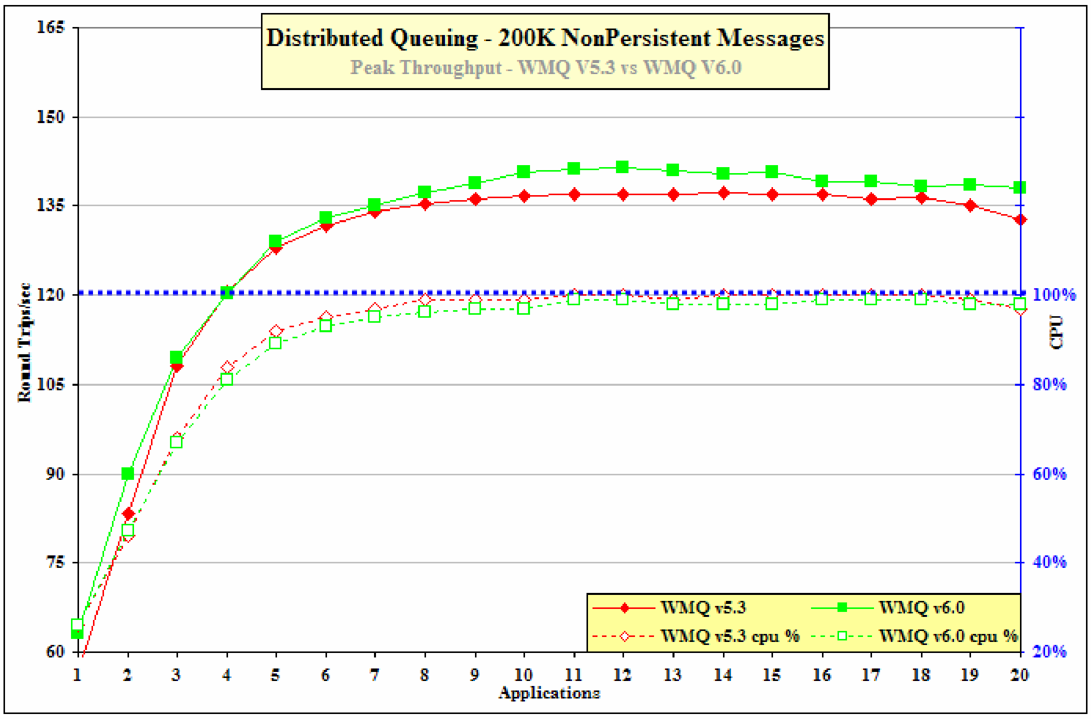


Figure 32 – 200K nonpersistent messages, distributed queuing

Figure 32 and **Table 20** show that the peak throughput of nonpersistent messages has increased by 3.1% (137 RT/s – 141 RT/s) comparing Version 5.3 to Version 6.0.

Test name: dqnp_200K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(12) **14** (20)	(137) **137** (133)	(0.1026) **0.1234** (0.1767)	(100%) **100%** (97%)
WebSphere MQ V6.0	**12** (14) (20)	**141** (140) (138)	**0.0996** (0.1182) (0.1694)	**99%** (98%) (98%)

Table 20 – 200K nonpersistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.*

3.3.3.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

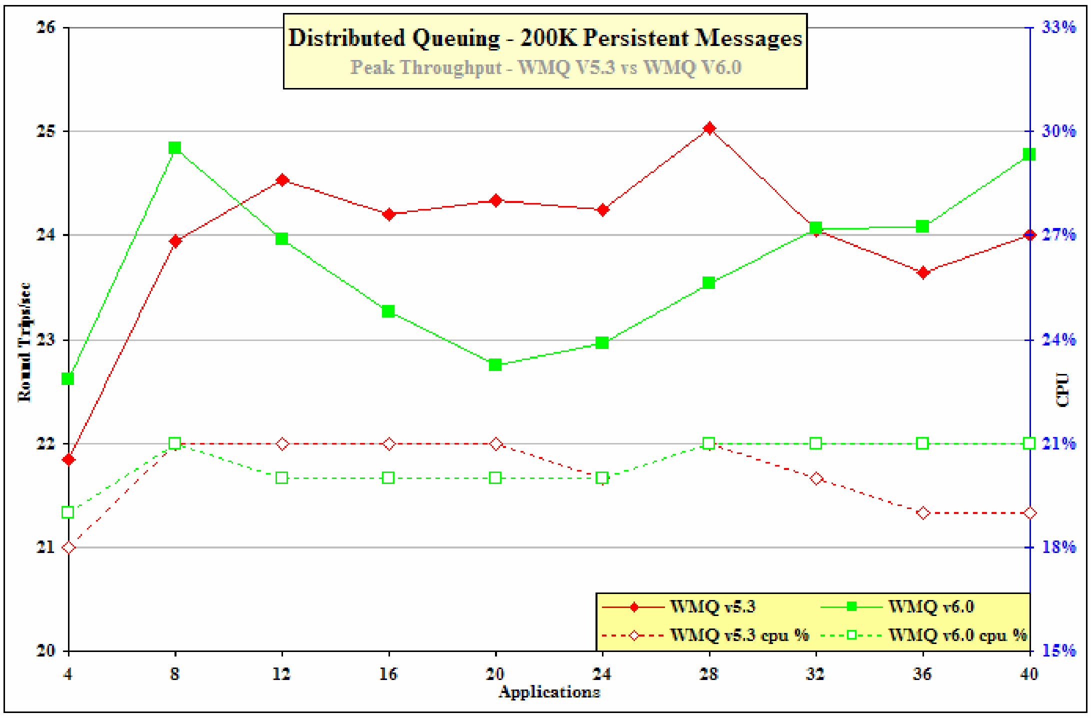


Figure 33 – 200K persistent messages, distributed queuing

Figure 33 and **Table 21** show that the peak throughput of nonpersistent messages is similar when comparing Version 5.3 to Version 6.0.

Test name: dqpm_200K	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(8) **28** (44)	(24) **25** (24)	(0.3636) **1.3406** (2.1666)	(21%) **21%** (19%)
WebSphere MQ V6.0	**8** (28) (40)	**25** (24) (25)	**0.3584** (1.3823) (1.8696)	**21%** (21%) (21%)

Table 21 – 200K persistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.4 2MB Messages

3.4.1 Local Queue Manager

Figure 34 and **Figure 35** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario.

3.4.1.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize = 25000000

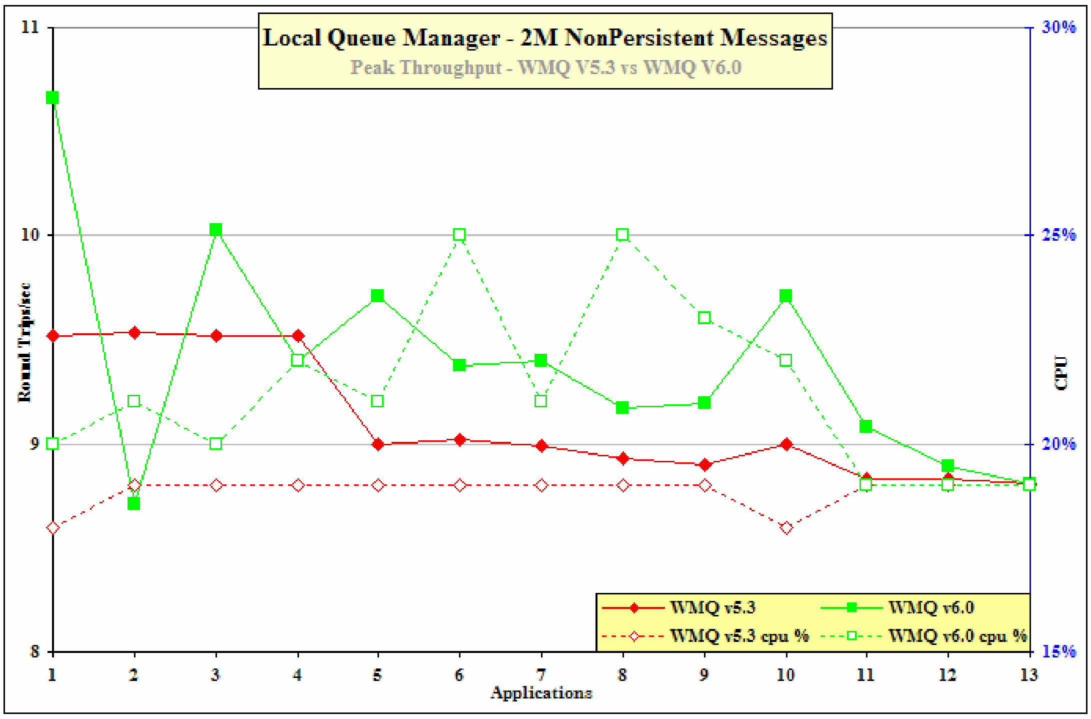


Figure 34 – 2M nonpersistent messages, local queue manager

Figure 34 and **Table 22** show that the peak throughput of nonpersistent messages is similar (10 RT/s – 11 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **local_np_2M**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	(1) **2** (13)	(10) **10** (9)	(0.1050) **0.2274** (1.7423)	(18%) **19%** (19%)
WebSphere MQ V6.0	**1** (2) (13)	**11** (9) (9)	**0.0938** (0.2452) (1.7412)	**20%** (21%) (19%)

Table 22 – 2M 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3*

3.4.1.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

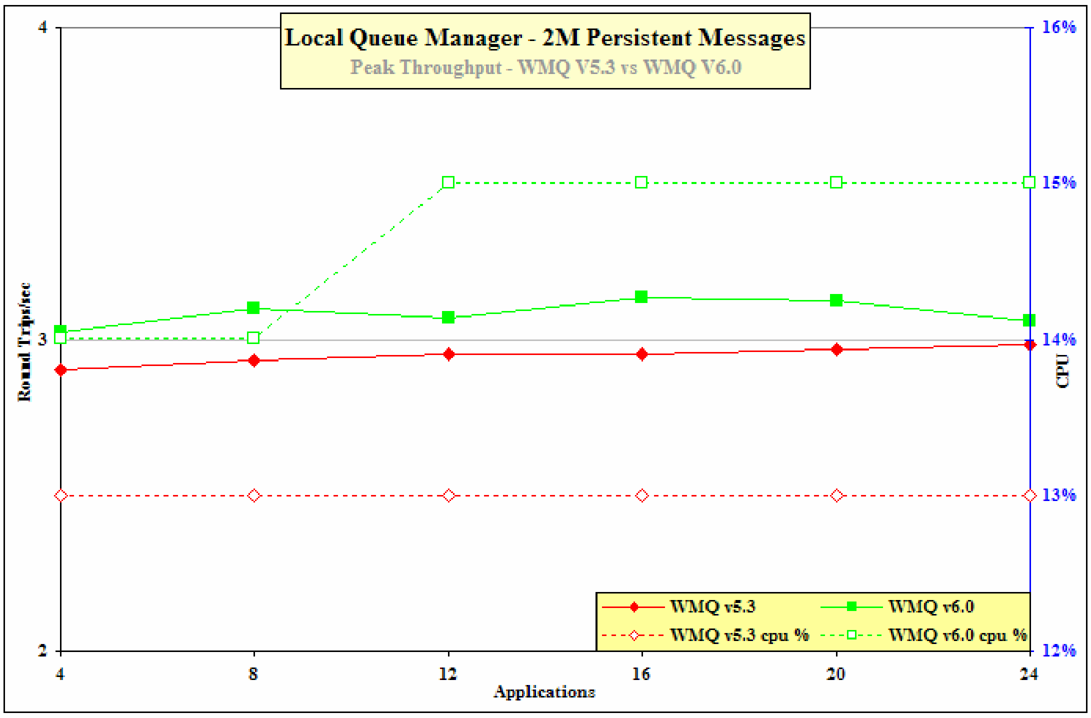


Figure 35 – 2M persistent messages, local queue manager

Figure 35 and **Table 23** show that the peak throughput of persistent messages is similar (3 RT/s – 3 RT/s) comparing Version 5.3 to Version 6.0.

Test name: **local_pm_2M**	**Apps**	**Round Trips/sec**	**Response time (s)**	**CPU**
WebSphere MQ V5.3	(16) **24**	(3) **3**	(6.1842) **9.4782**	(13%) **13%**
WebSphere MQ V6.0	**16** (24)	**3** (3)	**5.8641** (9.0213)	**15%** (15%)

Table 23 – 2M persistent 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.4.2 Client Channel

Figure 36 and **Figure 37** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client channel scenario.

3.4.2.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize = 25000000

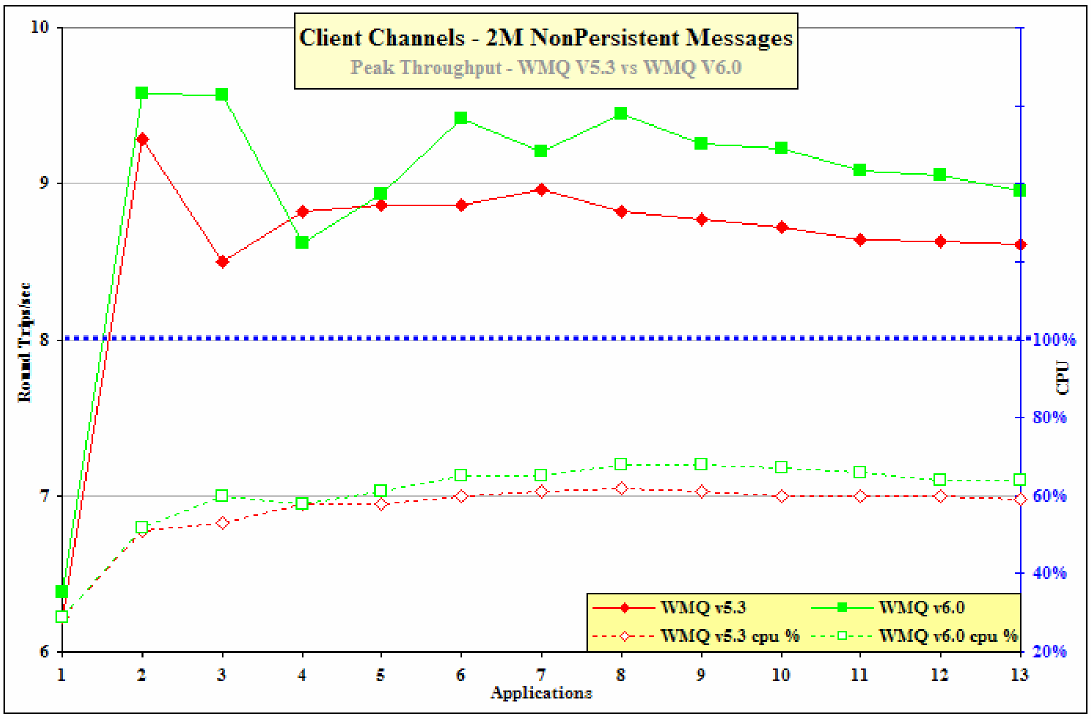


Figure 36 – 2M nonpersistent messages, client channels

Figure 36 and **Table 24** show that the peak throughput of nonpersistent messages is similar (9 RT/s – 10 RT/s) comparing Version 5.3 to Version 6.0.

Test name: clnp_2M	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**2** (13)	**9** (9)	**0.2422** (1.7748)	**51%** (59%)
WebSphere MQ V6.0	**2** (13)	**10** (9)	**0.2354** (1.7075)	(52%) **64%**

Table 24 – 2M nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.*

3.4.2.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

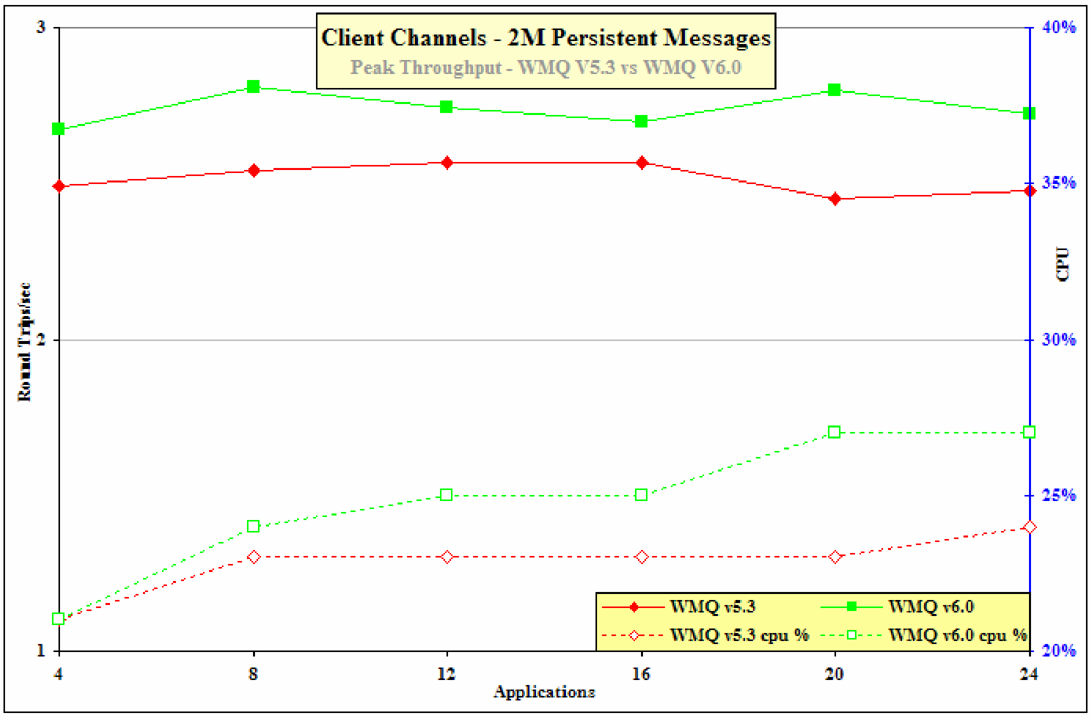


Figure 37 – 2M persistent messages, client channels

Figure 37 and **Table 25** show that the peak throughput of persistent messages is similar when comparing Version 5.3 to Version 6.0.

Test name: **clpm_2M**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	(8) **12** (24)	(3) **3** (2)	(3.4123) **5.2645** (10.9770)	(23%) **23%** (24%)
WebSphere MQ V6.0	**8** (12) (20)	**3** (3) (3)	**3.0953** (4.8228) (10.0768)	**24%** (25%) (27%)

Table 25 – 2M persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.4.3 Distributed Queuing

Figure 38 and **Figure 39** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario.

3.4.3.1 Nonpersistent Messages

Queue manager log configuration:

DefaultQBufferSize = 25000000

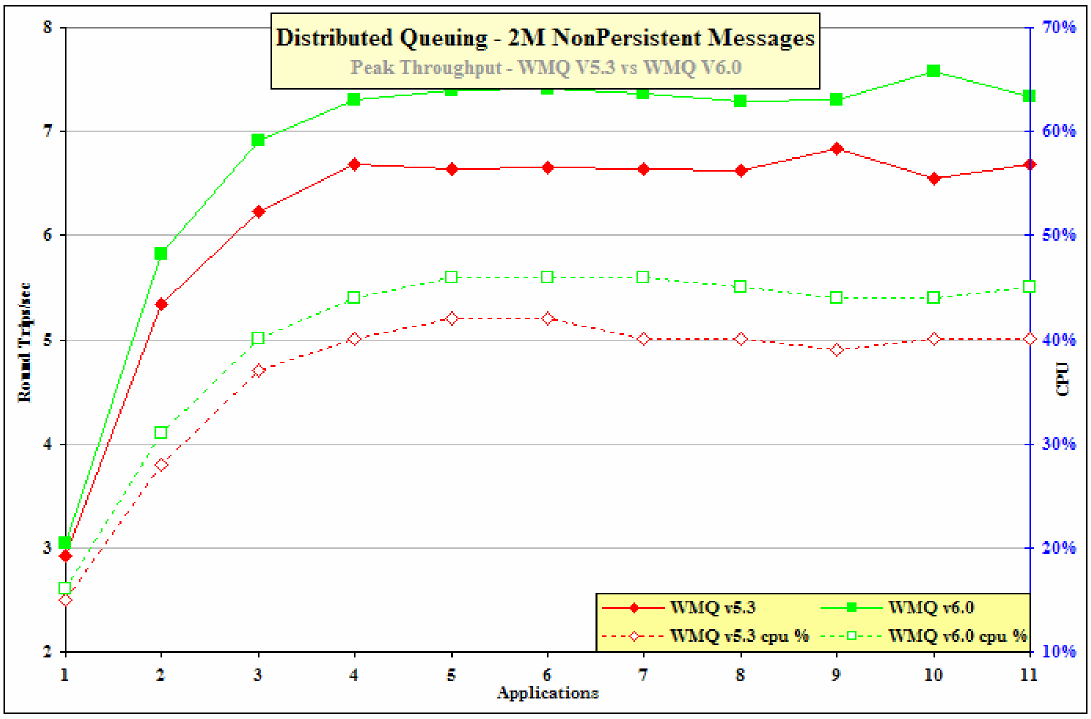


Figure 38 – 2M nonpersistent messages, distributed queuing

Figure 38 and **Table 26** show that the peak throughput of nonpersistent messages is similar (7 RT/s – 8 RT/s) comparing Version 5.3 to Version 6.0.

Test name: dqnp_2M	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**9** (10) (11)	**7** (7) (7)	**1.5476** (1.7818) (1.9553)	**39%** (40%) (40%)
WebSphere MQ V6.0	(9) **10** (12)	(7) **8** (8)	(1.4433) **1.5631** (1.8814)	(44%) **44%** (44%)

Table 26 – 2M nonpersistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

3.4.3.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

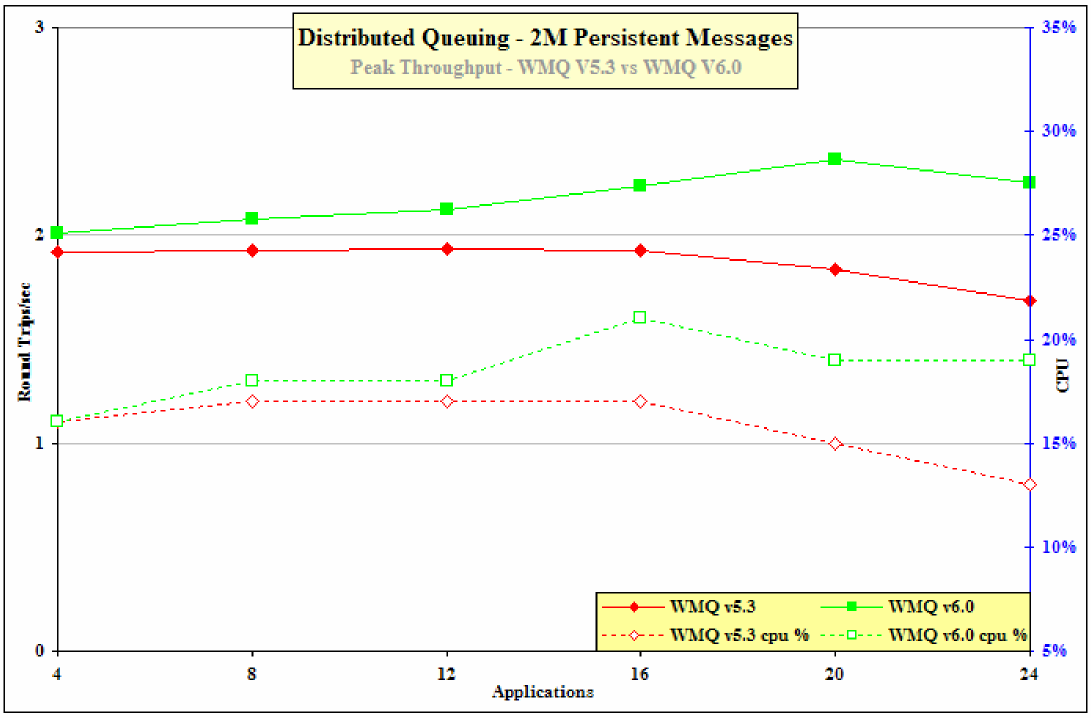


Figure 39 – 2M persistent messages, distributed queuing

Figure 39 and **Table 27** show that the peak throughput of nonpersistent messages is similar when comparing Version 5.3 to Version 6.0.

Test name: **dqpm_2M**	Apps	Round Trips/sec	Response time (s)	CPU
WebSphere MQ V5.3	**12** (20) (24)	**2** (2) (2)	**6.8394** (12.3247) (17.0313)	**17%** (15%) (13%)
WebSphere MQ V6.0	(12) **20** (24)	(2) **2** (2)	(6.3885) **9.8794** (12.7133)	(18%) **19%** (19%)

Table 27 – 2M persistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

4 Application Bindings

This report analyzes the rate that messages can be exchanged between a Requester (Driver) application and a Responder (Server) application. The other chapters use a 'Trusted' Requester and a 'Shared' Responder but this chapter looks at the effect of various combinations of application bindings for Requester and Responder programs.

	Requester	Responder
Normal	Trusted	Shared
Isolated	Isolated	Isolated
Trusted	Trusted	Trusted
NonTrusted	Shared	Shared

4.1 Local Queue Manager

Figure 40 and **Figure 41** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the local queue manager scenario.

4.1.1 Nonpersistent Messages

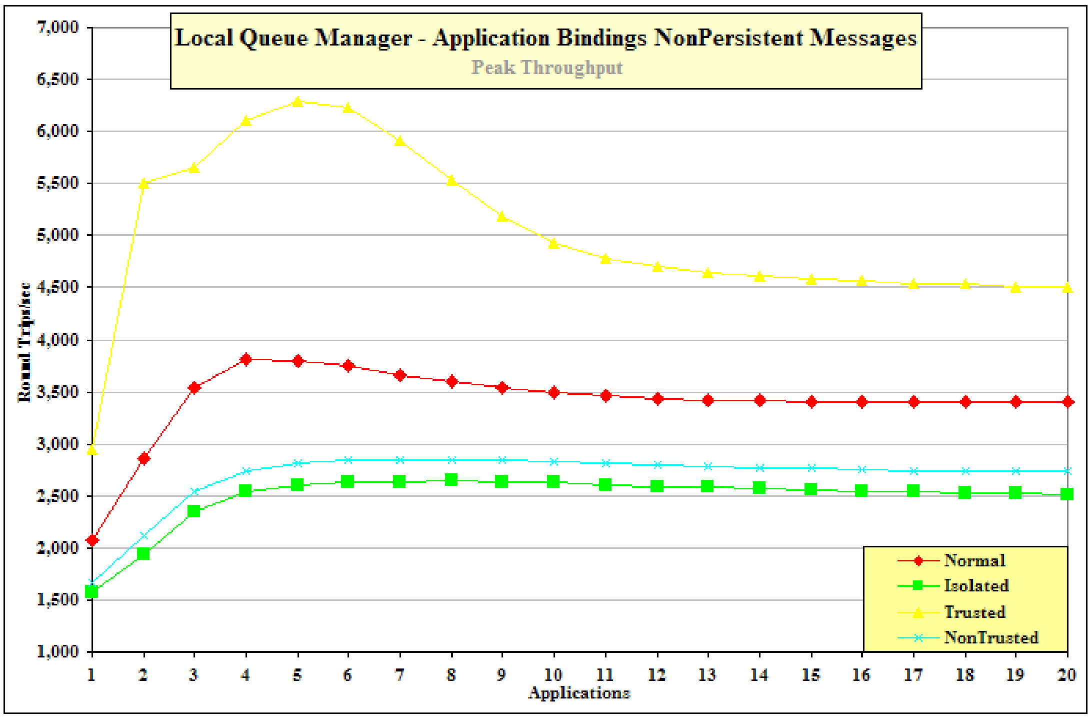


Figure 40 – Application binding, nonpersistent messages, local queue manager

Figure 40 and **Table 28** show that the peak throughput of nonpersistent messages when comparing Normal, Isolated, Trusted and NonTrusted bindings

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**4** (20)	**3,816** (3,396)	**0.0013** (0.0079)	**100%** (97%)
Isolated	**8** (20)	**2,651** (2,513)	**0.0036** (0.0098)	**100%** (99%)
Trusted	**5** (20)	**6,286** (4,510)	**0.0010** (0.0055)	**99%** (89%)
NonTrusted	**7** (20)	**2,850** (2,732)	**0.0030** (0.0090)	**100%** (100%)

Table 28 – Application binding, nonpersistent messages, local queue manager

4.1.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

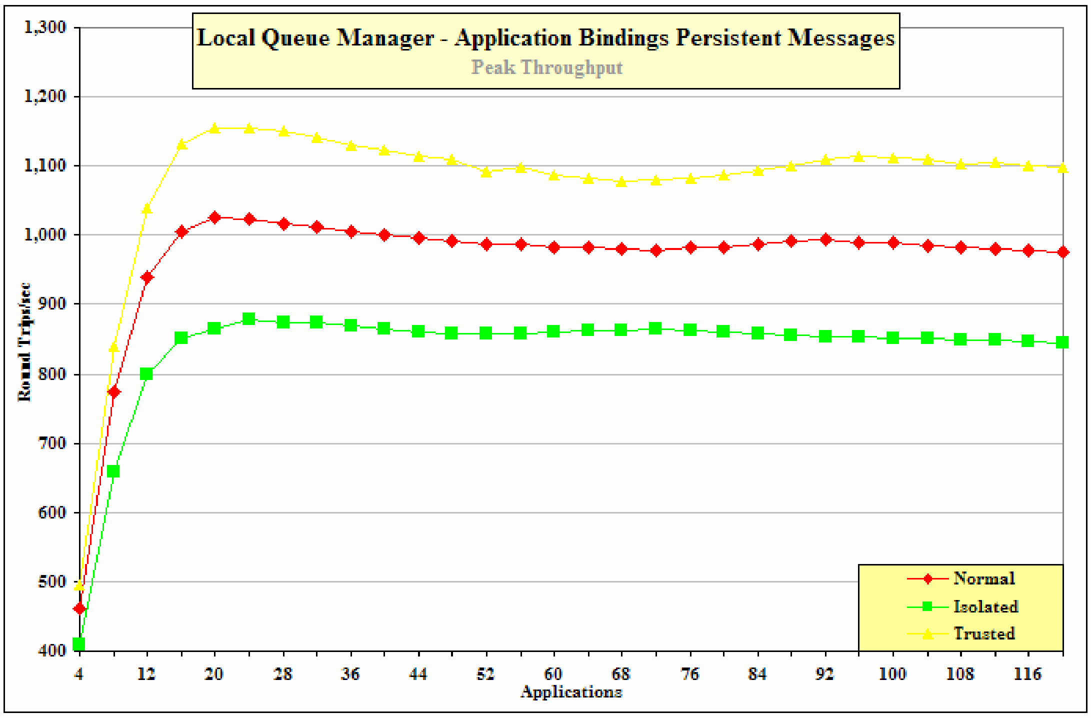


Figure 41 – Application binding, persistent messages, local queue manager

Figure 41 and **Table 29** show that the peak throughput of persistent messages when comparing Normal, Isolated and Trusted bindings.

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**20** (120)	**1,025** (977)	**0.0233** (0.1481)	**92%** (93%)
Isolated	**24** (120)	**877** (844)	**0.0331** (0.1732)	**98%** (98%)
Trusted	**20** (120)	**1,154** (1,098)	**0.0212** (0.1318)	**83%** (85%)

Table 29 – Application binding, persistent 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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

4.2 Client Channels

Figure 42 and **Figure 43** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the client channel scenario.

4.2.1 Nonpersistent Messages

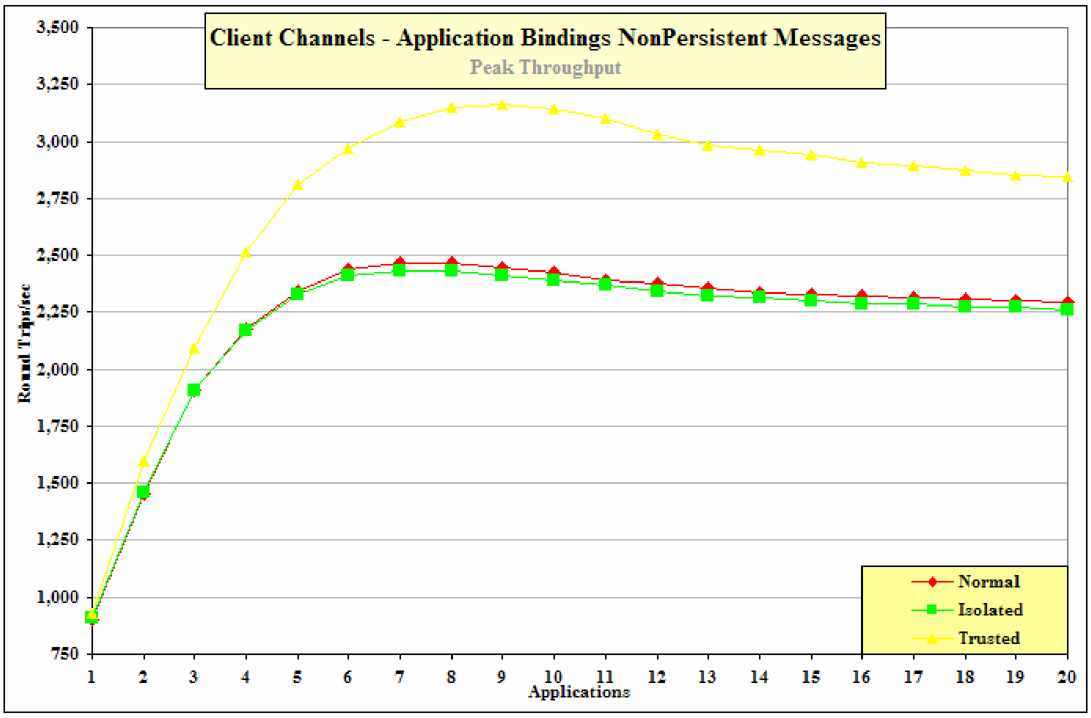


Figure 42 – Application binding, nonpersistent messages, client channels

Figure 42 and **Table 30** show that the peak throughput of nonpersistent messages when comparing Normal, Isolated and Trusted bindings.

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**7** (20)	**2,467** (2,291)	**0.0033** (0.0102)	**100%** (100%)
Isolated	**8** (20)	**2,433** (2,260)	**0.0038** (0.0101)	**100%** (99%)
Trusted	**9** (20)	**3,164** (2,844)	**0.0033** (0.0082)	**100%** (98%)

Table 30 – Application binding, nonpersistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

4.2.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

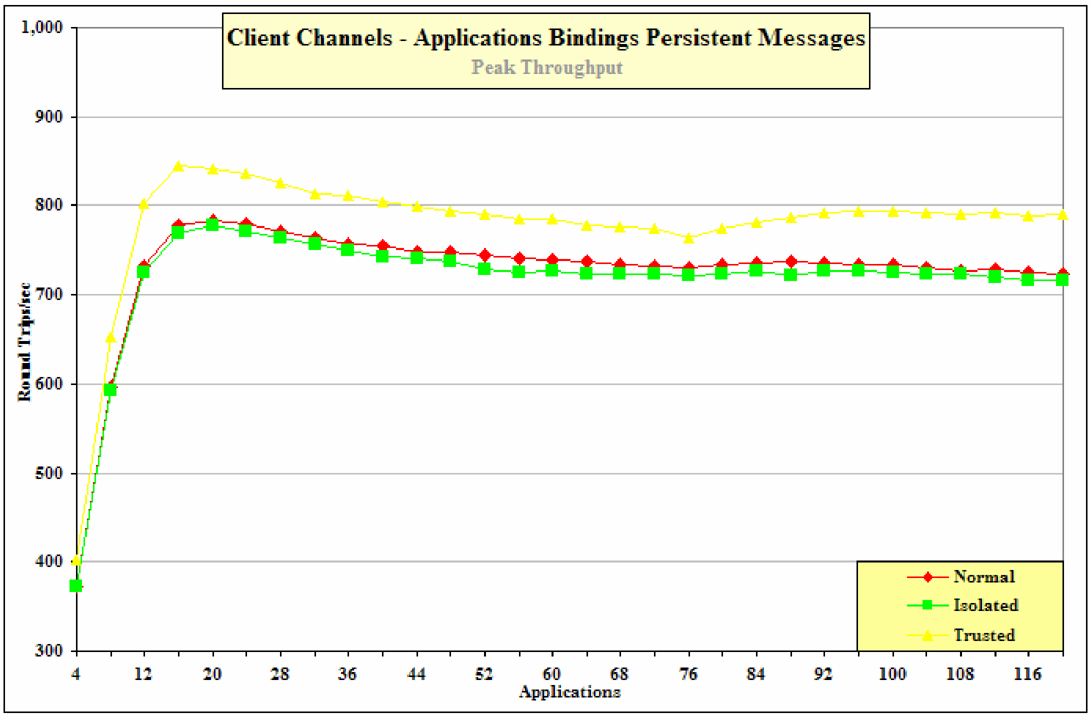


Figure 43 – Application binding, persistent messages, client channels

Figure 43 and **Table 31** show that the peak throughput of nonpersistent messages when comparing Normal, Isolated and Trusted bindings.

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**20** (120)	**782** (722)	**0.0301** (0.2009)	**96%** (97%)
Isolated	**20** (120)	**777** (717)	**0.0292** (0.2036)	**96%** (96%)
Trusted	**16** (120)	**844** (791)	**0.0216** (0.1834)	**90%** (94%)

Table 31 – Application binding, persistent messages, client channels

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

4.3 Distributed Queuing

Figure 43 and **Figure 44** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario.

4.3.1 Nonpersistent Messages

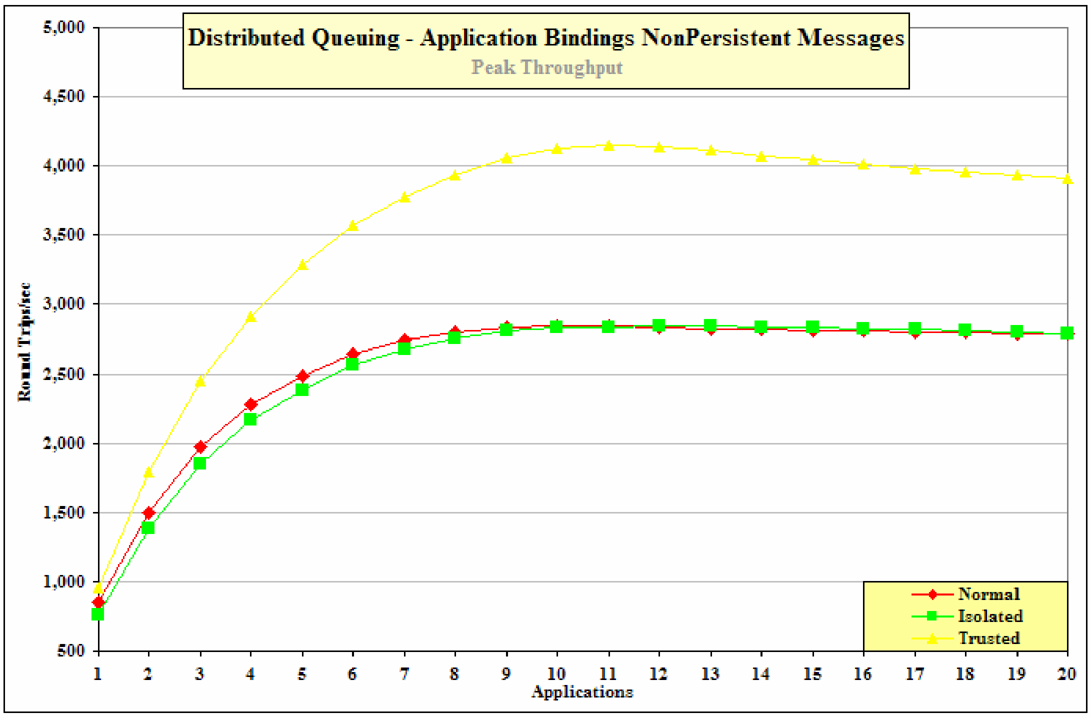


Figure 44 – Application binding, nonpersistent messages, distributed queuing

Figure 44 and **Table 32** show that the peak throughput of nonpersistent messages when comparing Normal, Isolated and Trusted bindings.

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**10** (20)	**2,844** (2,788)	**0.0041** (0.0085)	**100%** (100%)
Isolated	**12** (20)	**2,843** (2,790)	**0.0050** (0.0086)	**100%** (100%)
Trusted	**11** (20)	**4,153** (3,910)	**0.0033** (0.0061)	**100%** (99%)

Table 32 – Application binding, nonpersistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

4.3.2 Persistent Messages

Queue manager log configuration:

LogPrimaryFiles=4, LogFilePages=16384, LogBufferPages=512

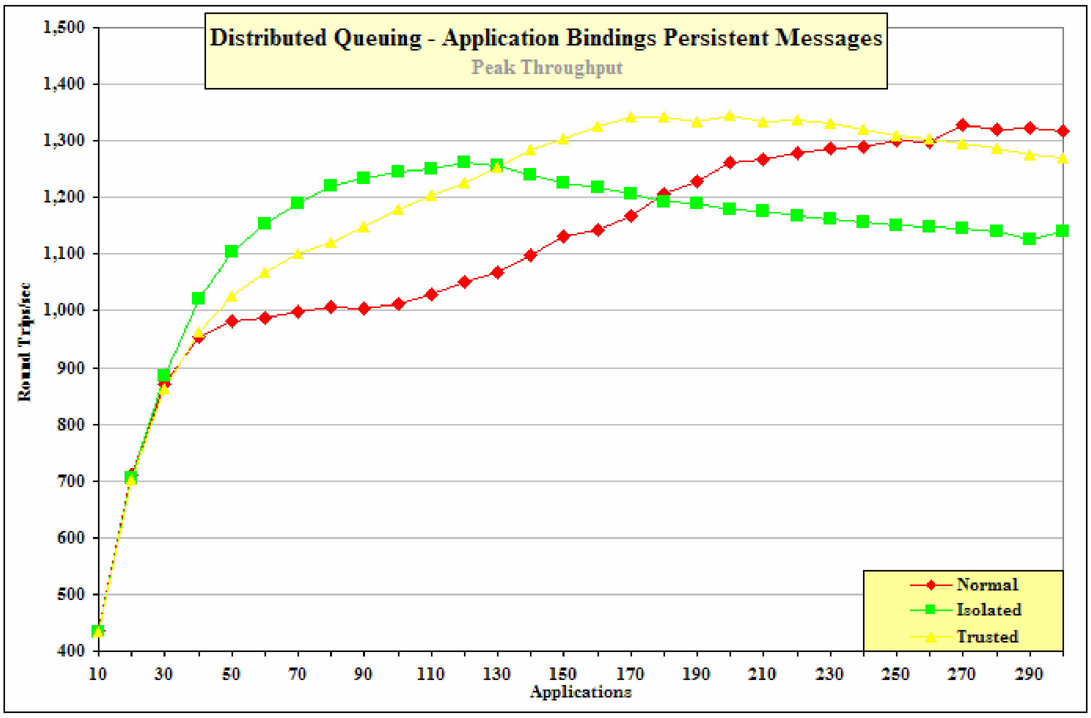


Figure 45 – Application binding, persistent messages, distributed queuing

Figure 45 and **Table 33** show that the peak throughput of nonpersistent messages when comparing Normal, Isolated and Trusted bindings.

Test	Apps	Round Trips/sec	Response time (s)	CPU
Normal	**270** (300)	**1,328** (1,317)	**0.2223** (0.2588)	**93%** (94%)
Isolated	**120** (300)	**1,262** (1,140)	**0.1113** (0.3130)	**92%** (89%)
Trusted	**200** (300)	**1,344** (1,271)	**0.1782** (0.2763)	**95%** (93%)

Table 33 – Application binding, persistent messages, distributed queuing

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 achieve the peak throughput. The numbers in brackets are included in the table to provide meaningful comparison between WebSphere MQ V6.0 and Version 5.3.

5 Short Sessions

The previous chapters in this report only reported on steady state messaging that does not include any session setup and termination function. This chapter specifically bracket groups of five MQPUT/MQGET pairs with MQCON/MQDISC and MQOPEN/MQCLOSE calls so a comparison of this overhead can be seen.

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
- wait one second
- closes both queues
- disconnects from the queue manager

5x

"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 46** for an illustration).

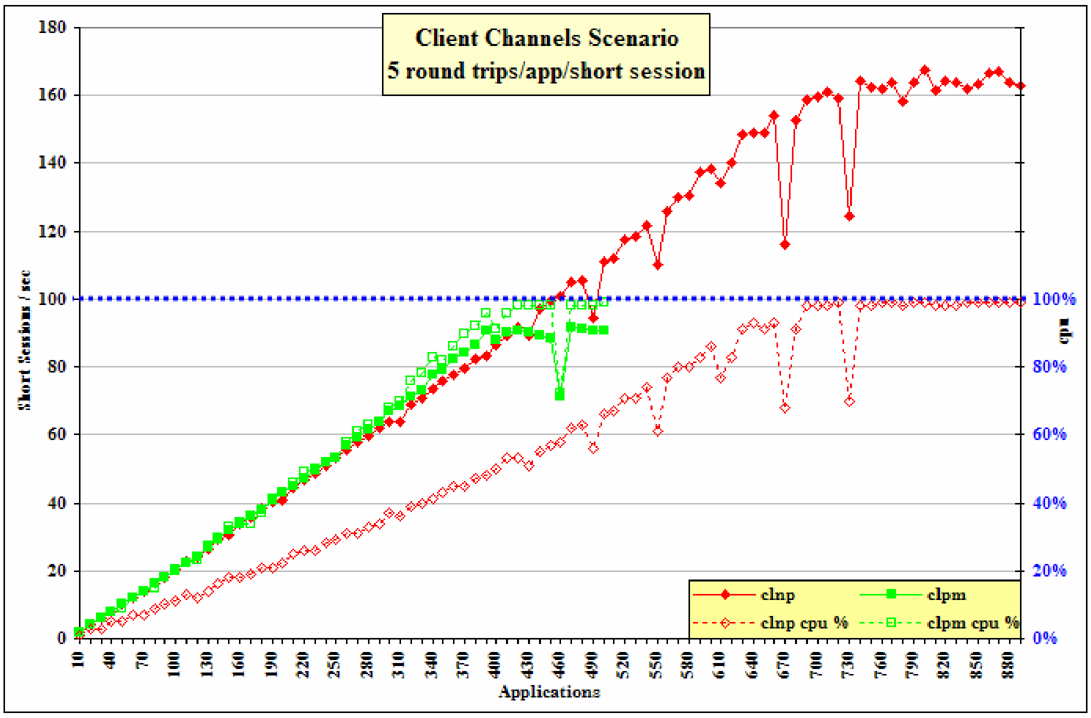


Figure 46 – Short sessions, client channels

Test name	Apps	Round Trips/sec	Short Sessions per second	Response Time (s)
clnp_r3600	**1500**	**1499**	▨	**0.3229**
clnp_ss	**750** (800)	**813** (837)	**162** (167)	**0.6848** (1.2031)
clpm_r3600	**650**	**649**	▨	**0.2654**
clpm_ss	**420** (470)	**453** (457)	**90** (91)	**0.6135** (1.2286)

Table 34 – Short sessions, 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 are for WebSphere MQ V6.0 with a round trip response time of less than one second. The smaller figures in brackets show maximum throughput regardless of response time. 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. INETD listener has a significantly smaller capacity because of the need to create a new process for every client.

6 Performance and Capacity Limits

6.1 Client channels – capacity measurements

The measurements in this section are intended to test the maximum number of client channels into a server queue managers with a messaging rate of 1 round trip per client channel per *minute.* Measurements are also made with smaller number of Client channels where the message insertion rate is increased until the system gets congested. This information is intended to be useful to the reader sizing a system with similar scenarios.

Queue manager configuration for client channels capacity tests:

MaxChannels=50000

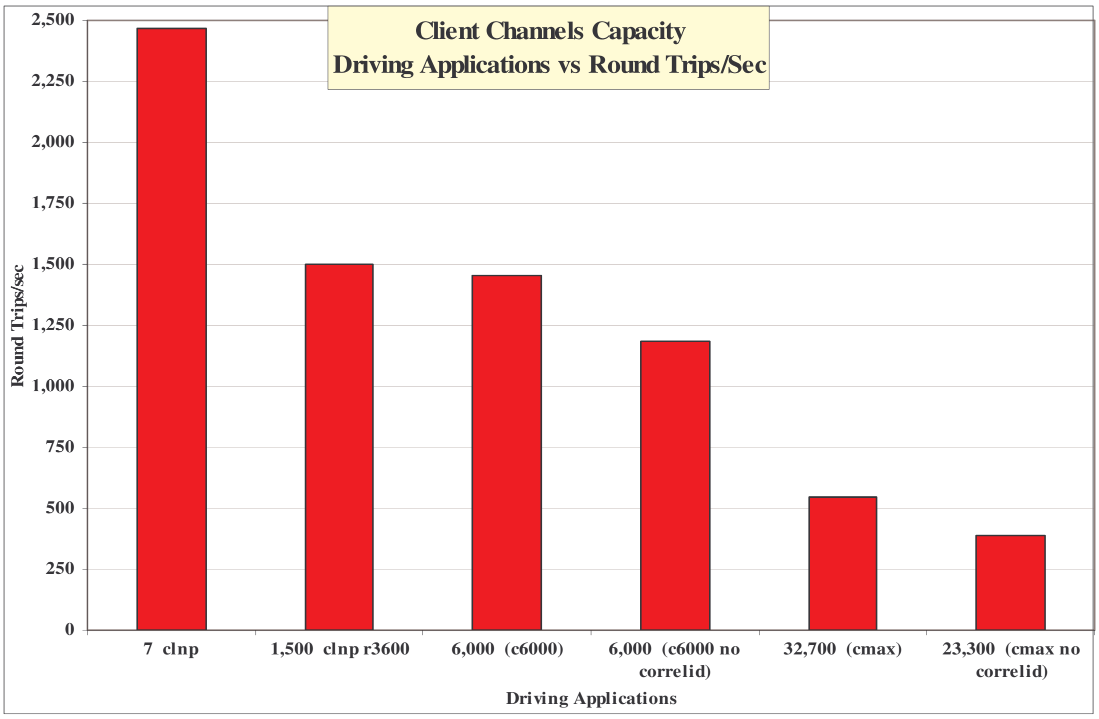
Test name:	Apps	Rate/app/hr	Round Trips/sec	Response time (s)	CPU
clnp	**7**	**n/a***	**2,467**	**0.0033**	**100%**
clnp_r3600	**1,500**	**3600**	**1,499**	**0.3229**	**85%**
clnp_c6000	**6,000**	**870**	**1,452**	**0.5709**	**77%**
clnp_c6000_no_correllid	**6,000**	**710**	**1,183**	**0.3562**	**91%**
clnp_cmax	**32,700**	**60**	**545**	**0.1189**	**6%**
clnp_cmax_no_correllid	**23,300** (17000)	**60**	**387** (283)	**0.3136** (0.1209)	**69%** (15%)

Table 35 – Capacity measurements, client channels

* *There was no delay between the response to the previous message and the insertion of the next message with 7 clients.*

The clnp_cmax_no_correllid results are shown when there is no paging (@ 17000) and again when the response time has not been significantly degraded due to paging because there is spare CPU available. The effect of the number of client channels on maximum message throughput is shown in **Figure 47** below.

Figure 47 – Effect of number of client channels on round trips



Test name:	Apps	Swap	Free
clnp_cmax	**32,700** (1,000)	**4926MB (139K/App)** (434MB)	**120 KB/App**
clnp_cmax_no_correllid	**23,300** (500)	**4854MB (202K/App)** (376MB)	**179KB/App**

Table 36 – Client capacity, memory utilisation

Note: The table above show the swop memory measured at the given number of driving applications. The swop and free memory cost is the additional cost per driving application (in this test scenario this relates to the cost of an MQI-client connection on the server machine).

The difference between this pair of measurements is the clnp_cmax uses a Get by Correlation_Id from a common reply queue for all the clients whereas the other case has a separate reply queue per client. Each additional Client needs a thread in the AMQRMPPA process and this accounts for most of the 120K bytes. Using a separate queue per client needs additional shared memory per client as well as some more pages from the Free pool. The Operating System page manager will need about 75% of this in real memory to ensure that system response time is not degraded

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*. 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 useful to the reader sizing a system with 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 Trips/sec	Response time (s)	CPU
dqnp	10	n/a*	2,844	0.0041	100%
dqnp_r3600	2,700	3600	2,700	0.1178	97%
dqnp_q1000	1,000	6,430	1,785	0.0698	87%
dqnp_qmax	10,000	60	167	0.1185	4%

Table 37 – Capacity measurements, server channels

* *There was no delay between the response to the previous message and the insertion of the next message with 10 driving applications.*

The effect of the number of server channel pairs on maximum message throughput can be seen in **Figure 48** below.

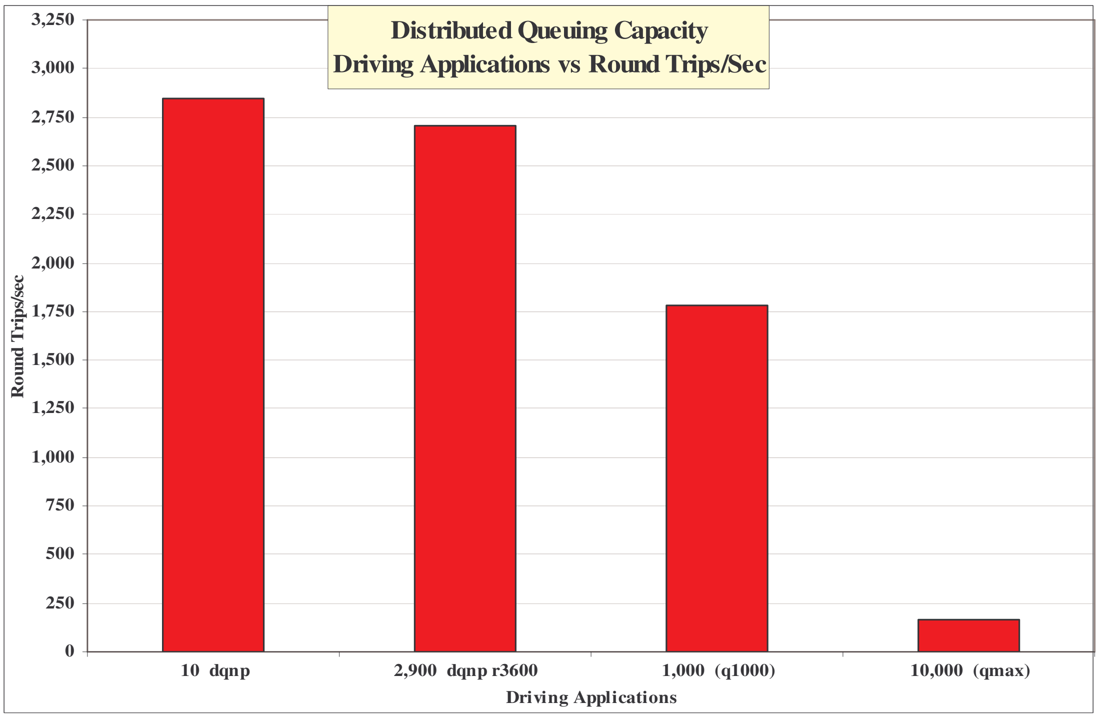


Figure 48 – Effect of number of server channels on round trips

The dqnp and dqnp_r3600 both used a total of 2 pairs of Sender/Receiver pairs of channels between queue managers while the dqnp_q1000 and dqnp_qmax used a pair of channels per application. The dqnp_q1000 shows the reduced throughput experienced when 1000 queue mangers are connected into a central hub and the following table shows the storage on the central hub.

Test name:	Apps	Swap	Free
dqnp_qmax	10,000 (500)	4779MB (383K/App) (954MB)	340 KB/App

Table 38 – DQ capacity, memory utilisation

Note: *The table above show the swap memory measured at the given number of driving applications. The swap and free memory cost is the additional cost per driving application (in this test scenario this relates to the cost of an MQI-Sender/Receiver pair of channels connected on the server machine).*

It can be observed that the amount of memory needed to support a DQ channel is approximately double that needed to support a client channel. Each DQ application needs both a Sender and Receiver channel as well as a Transmit queue whereas a Client connection just needs one channel.

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 V6.0; some of these can be applied to Version 5.3. 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 and Persistent Queue Buffer

The default nonpersistent queue buffer size is 64K per queue and the default persistent is 128K per queue. These can be increased to 1MB using the TuningParameters stanza and the *DefaultQBufferSize* and DefaultPQBufferSize parameters. (For more details see SupportPac MP01: MQSeries – Tuning Queue Limits). Increasing the queue buffer provides the capability to absorb peaks in message throughput at the expense of real storage. Once these queue buffers are full, the additional message data is given to the file system that will eventually find its way to the disk. Defining queues using large nonpersistent or persistent 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 queue buffers are 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 *DefaultPQBufferSize*. The value is taken from the TuningParameters stanza in use by the queue manager when the queue was defined. 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

The log buffer is a piece of main memory where the log records are appended so that multiple log records can be written to disks together. The default size of the log buffer is 128 pages with a maximum size of 4096 pages. To improve persistent message throughput the *LogBufferPages* should be increased to *512* x 4K pages = 2MB, or larger. *LogFilePages* (i.e. `crtmqm -lf <LogFilePages>`) defines the size of one physical disk extent and should be configured to a large size, for example: *16384* x 4K pages = 64MB, with the maximum size being 65535 pages. The number of *LogPrimaryFiles* (i.e. `crtmqm –lp <LogPrimaryFiles>`) should be configured to a large number and the maximum number of Primary plus Secondary extents is 255(Windows) and 511(UNIX). The cumulative effect of this tuning will:

- Improve the throughput of persistent messages (permitting a possible 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 a 2MB queue manager log but processing of large messages will be helped by these enhanced limits. 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 T*hread*, Standard or F*astpath*?

Threaded channels are used for all the measurements in this report ('runmqlsr', and for server channels an MCATYPE of 'THREAD') 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 rather than 'inetd', 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.

Note: 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 (Shared or Isolated)* 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 typically 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 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 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 messages are being placed on a queue quicker than they can be removed. The first effect is that messages begin to fill the 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 queue buffer, it may not be possible to avoid a performance degradation.

Processing persistent 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 while outside of syncpoint control, the queue manager must wait for each message to be written to the log before returning control to the application.

Only one log record per queue can be written to the disk per log I/O when processing messages outside of syncpoint. This is not a bottleneck when there are a lot of different queues being processed. When there are a small number of queues being processed by a large number of parallel application threads, it is a bottleneck. By changing all the messages to be processed inside syncpoint, the bottleneck is removed because multiple log records per queue can share the same log I/O for messages processed within syncpoint.

A typical triggered application follows the performance profile of a short session . 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. 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 flexibility 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 Virtual Memory, Real Memory, & Paging

Systems require sufficient real memory to hold the working set otherwise paging will break the response time expectations.

- Virtual memory enables the program to address much larger amount of memory than exists as real memory.
- Real memory is the physical memory (or RAM) currently installed in the machine.
- Paging is the process of managing program access to virtual storage pages not currently resident in main memory. It locates the required page frame from auxiliary storage (disk), selects a page frame in real memory that will hold this page, copies the contents of this outgoing page frame to auxiliary storage, and retrieves the requested incoming page contents from auxiliary storage.

A simple approach is to ensure that the virtual memory of the application system does not exceed the available real memory since all memory requests will be met from the current free memory. VMSTAT reports on 'in use' and 'free' memory as seen by the operating system page manager.

WebSphere MQ uses a significant amount of memory for each Queue Manager and Channel.

7.3.1 Queue Manager

Starting a MQ Queue manager generated using default values increases the AVM and reduces the FRE by 32M bytes.

7.3.2 Channels

Channels can be started by using the INETD or the RUNMQLSR listener. INETD initiated channels use between 5 and 10 times more memory than RUNMQLSR channels so the rest of this section focuses on RUNMQLSR channels.

7.3.3 Client Channels

Each MQ client channel uses between 264K - 410K bytes for processing 2K byte messages depending on traffic rate (Chapter 6 of the MQ V6 Performance reports provides an estimate of the storage needed when clients either share a predefined queue with other clients or have a dynamic queue per client). 100K byte messages will use up to 700K bytes per client.

7.3.4 Server – Server Channels

Each interconnected queue manager has a pair of uni-directional channels for sending and receiving messages. The storage consumed is the same as 2 client channels plus a predefined queue (Transmission queue).

Three other aspects of storage consumption depend on type of 'Reply-Queue', MQIBINDTYPE, and BufferLength.

7.3.5 Reply Queue

The Queue from which the client retrieves the message can be a predefined Queue (350K bytes) probably shared among multiple clients who get messages by Correlation-id or a model (dynamic) queue (60K bytes) that is used only by one client. The model queue memory can grow by 128K bytes when more than 128K bytes of Persistent messages are held in the queue and by 192K bytes when more than 192K bytes of non persistent messages are held in the queue. This memory is not shrunk back to the underlying 60K bytes for model queues.

7.3.6 BufferLength

The AMQRMPPA process contains a thread per connected client. The BufferLength parameter of the MQGet is also used to allocate a long term piece of storage of this size in which the message is held before being retrieved by the client. If the size of the arriving messages cannot be predicted then the application should provide a buffer than can deal with 90% of the messages and redrive the MQGET after return code **2080 (X'0820') MQRC_TRUNCATED_MSG_FAILED** by providing a larger BUFFER for retrieving this particular message. There is a mechanism to gradually reduce the size of the storage in AMQRMPPA if the recent BufferLength size is significantly smaller than previous BufferLength.

7.3.7 MQIBINDTYPE

MQIBINDTYPE=FASTPATH will cause the channel to run ‘Trusted’ mode. Trusted applications do not use a thread in the Agent (AMQZLLA) process. This means there is no IPC between the Channel and Agent because the Agent does not exist in this connection. If the channel is run in STANDARD mode then any messages passed between the channel and agent will use IPCC memory (size = BufferSize with a maximum size of 1MB) that is dynamically obtained and only held for the lifetime of the MQGet. Standard channels each require an additional 80K bytes of memory. As the message rate increases, there will be more IPCC memory used in parallel.

The power of the machine used to process a workload needs to handle the peaks of troughs. Customers may specify a daily workload but this number cannot be divided by the number of seconds in a day to find the necessary system configuration. The peak hourly rate cannot be divided by 3600 because the peak rate per second will probably be 2-3 times higher. The system must process these peak loads without building up a backlog of queued work. It is important to prevent the queue depths increasing because they will occupy memory from the 'fre' pool or be spilled out to disk. Over commitment of real memory is handled by the page manager but sudden large jumps (storms) possibly due to queues becoming deep can cause the throughput to break down completely if the page manager chooses too much working set memory to be paged. Gradual over commitment enables the page manager to shuffle out those pages that are not part of the working set.

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

The MQI applications use 32 bit libraries for MQ V53 and 64 bit libraries for MQ V6.

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 driving applications (Requesters) for Local and DQ tests used Trusted bindings. 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 6**.

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 20, 6, 6 threads for processing nonpersistent messages with Local, Client, and DQ applications, and 30, 60, 10 threads to process persistent messages with Local, Client, and DQ applications. Each server thread performed the sequence of actions as outlined in the test scenario illustrations in the '**Performance Headlines**' starting on **page 6**.

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

Sun Ultra-80 420R:	Server system (device under test)
Model:	Z801 0000022003
Processor:	450MHz UltraSparc-II
Architecture:	4-way SMP
Memory (RAM):	4GB
Disk:	2 Internal Ultra2 SCSI (18.2GB ea. 1 O/S, 1 swap)
	2 External Ultra2 SCSI (18.2GB ea. 1 queue, 1 log)
Network:	1GBit Ethernet

IBM S80:	Driving applications machine
Model:	7017-S80
Processor:	375MHz PowerPC RS64-III
Architecture:	24-way SMP
	IBM SSA 160 SerialRAID Adapter
Memory (RAM):	32GB
Disk:	2 Internal 16Bit LVD SCSI (9.1GB ea. 1 O/S, 1 O/S + swap)
	3 SSA Logical disks
	(1 Physical SSA160, 9.1GB, 1 swap, 1 queue, 1 log)
Network:	1GBit Ethernet

Sun Sun-Fire 3800:	Driving applications machine (not used)
Model:	Z801 0000024977
Processor:	750MHz UltraSparc-III
Architecture:	4-way SMP
Memory (RAM):	4GB
Disk:	2 Internal Ultra2 SCSI (36GB ea. 1 O/S, 1 swap)
	2 External Ultra2 SCSI (36GB ea. 1 queue, 1 log)
Network:	1GBit Ethernet

8.3 Software

Solaris O/S:	SunOS Version 5.8
MQSeries:	Version 6.0 (B.11.600.???), Version 5.3 (B.11.530.???)
Compiler:	C for AIX Compiler, Version 6

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.