WebSphere MQ for IBM i V7.0 - Performance Evaluations

Version 1.1

January 2009

Eric Stec, Dale Weber, Craig Stirling, 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, January 2009

This edition applies to *WebSphere MQ for IBM i V7* (and to all subsequent releases and modifications until otherwise indicated in new editions).

Notices

DISCLAIMERS
The performance data contained in this report were measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends on the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION
The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS
The information set forth in this report could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

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

wanting to understand the performance characteristics of *WebSphere MQ for IBM i V7*. The information is not intended as the specification of any programming interface that is provided by WebSphere. It is assumed that the reader is familiar with the concepts and operation of WebSphere MQ V7.

LOCAL AVAILABILITY
References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES
Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS AND SERVICE MARKS
The following terms used in this publication are trademarks of International Business Machines Corporation in the United States, other countries or both:

- IBM
- WebSphere
- DB2

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

Other company, product, and service names may be trademarks or service marks of others.

EXPORT REGULATIONS
You agree to comply with all applicable export and import laws and regulations.

Preface

Target audience

This SupportPac is designed for people who:

- Will be designing and implementing solutions using WebSphere MQ for IBM i.
- Want to understand the performance limits of WebSphere MQ for IBM i V7.0.
- Want to understand what actions may be taken to tune WebSphere MQ for IBM i.

The reader should have a general awareness of the IBM I 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 familiarize 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

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).

An IBM i model 550 (4-way CPU 1.9ghz Power 5) with 32GB of RAM was used as the device under test.

An IBM xSeries 365 (4-way CPU 3.0ghz Intel Pentium 4 Xeon) with 4GB of RAM was used as the driver for all tests.

How this document is arranged

Performance Headlines

Page: 1

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

Page: 18

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: 39

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

Short & Long Sessions

Page: 45

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

Page: 47

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

Page: 49

This section contains performance tuning recommendations for WebSphere MQ 7.0 on IBM i.

Measurement Environment

Page: 52

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: 53

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

CONTENTS

TABLES

FIGURES

1 Overview

WebSphere MQ V7.0 on IBM i has similar performance characteristics to the V6 product. The comparisons in this report show that throughput is similar overall (for local, client and distributed queuing) when the clients are running in V6 compatibility mode. The default enhanced client support that provides heartbeating, enhanced reliability and multiplexing degrades client benchmarks by 5-15%.

There are new functions in V7 that provide enhanced performance to applications that are able to use them. These include Asynchronous Puts, Read-ahead, Properties, and selectors, but they are not covered in this document.

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

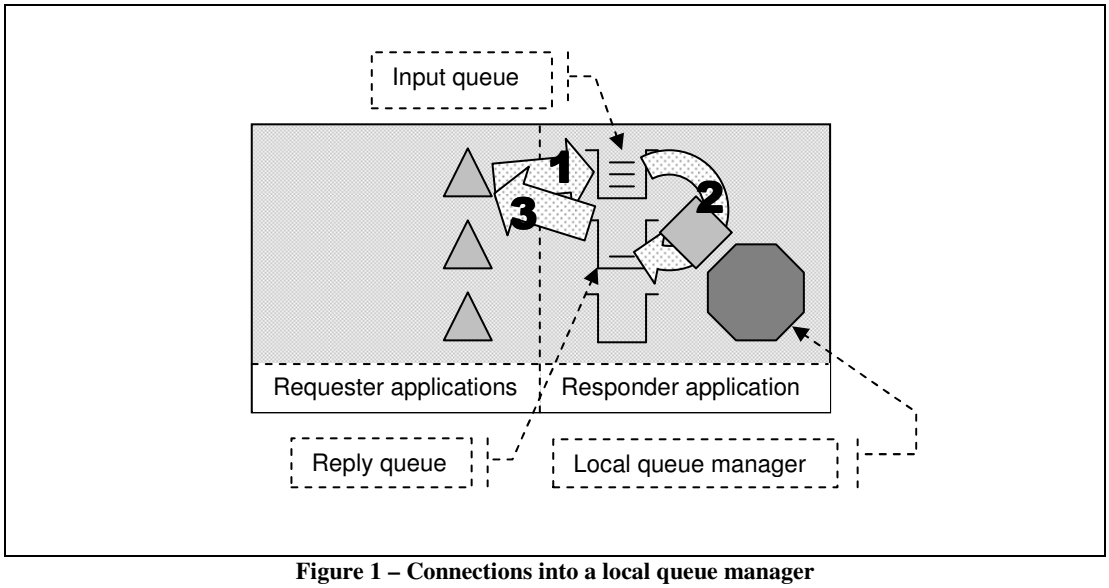


Figure 1 – Connections into a local queue manager

1) The Requester 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 Requester application then waits indefinitely for a reply to arrive on the common reply queue.

2) The Responder 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 Requester 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.

Application Bindings of the Responder program are 'Shared' and the Requester program is normally 'Trusted' except in the 'non-trusted' scenario where both programs use 'shared' bindings.

2.1.1 Nonpersistent Messages – Local Queue Manager

Figure 2 , **Figure 2a** 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 V7.0 compared to Version 6.0.

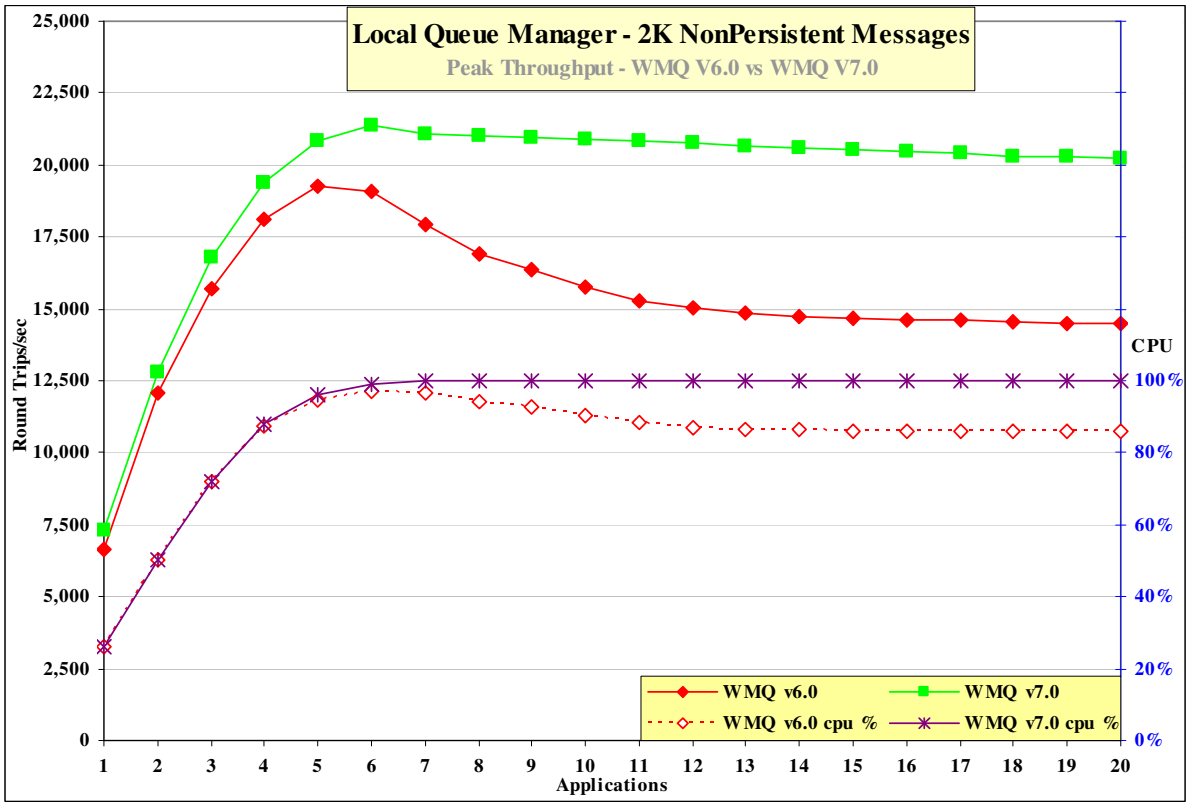


Figure 2 – Performance headline, nonpersistent messages, local queue manager

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

Figure 2 and **Table 1** shows that the maximum throughput of nonpersistent messages has improved by 11.0% comparing Version 6 to Version 7.

Test name: **local_np**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**5** (6)	**19272** (19093)	**0.0003** (0.0004)	**95%** (97%)
WebSphere MQ V7.0	(5) **6**	(20831) **21397**	(0.0003) **0.0003**	(96%) **99%**

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

2.1.2 Nonpersistent Messages – NonTrusted – Local Queue Manager

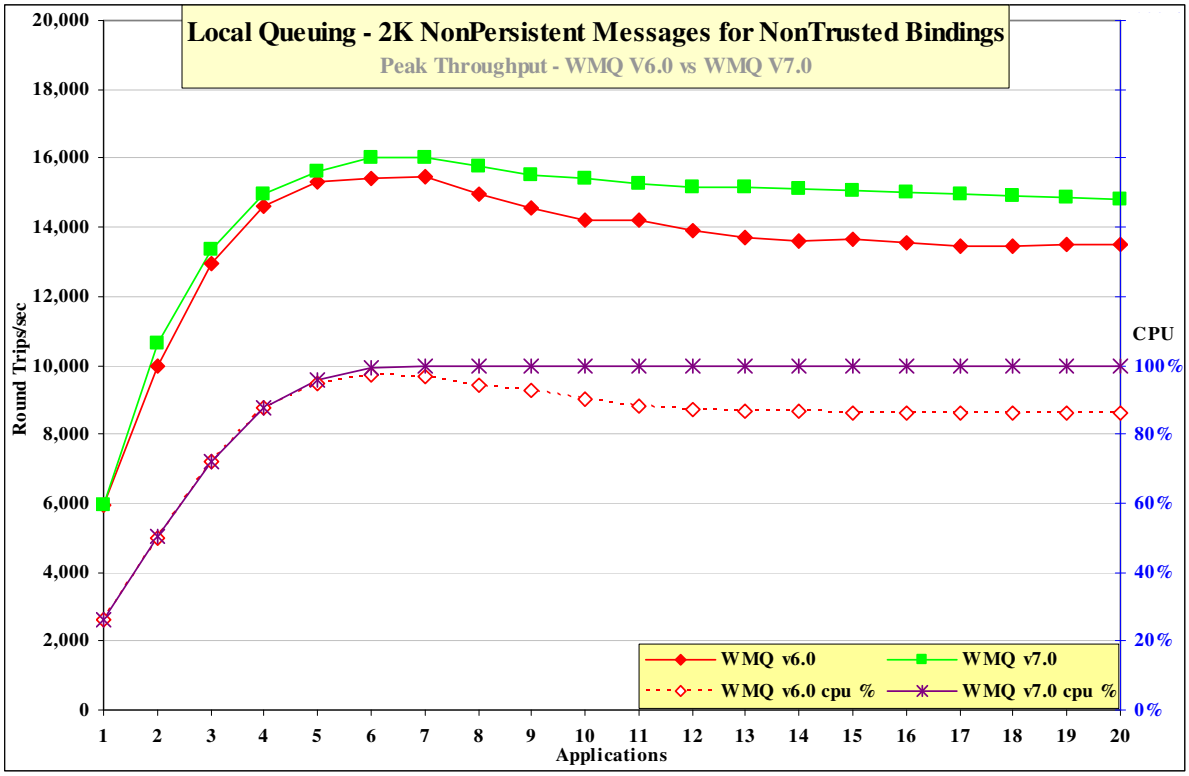


Figure 2a

Test name: **local_np_nt**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V7.0	(6) **7**	(15423) **15466**	(0.0005) **0.0006**	(99%) **99%**
WebSphere MQ V6.0	**6** (7)	**16030** (16030)	**0.0005** (0.0006)	**100%** (97%)

The above chart and table show that the maximum throughput of nonpersistent messages when the requester and responder both use shared bindings has improved by 3.7% comparing Version 6 to Version 7.

2.1.3 Persistent Messages – Local Queue Manager

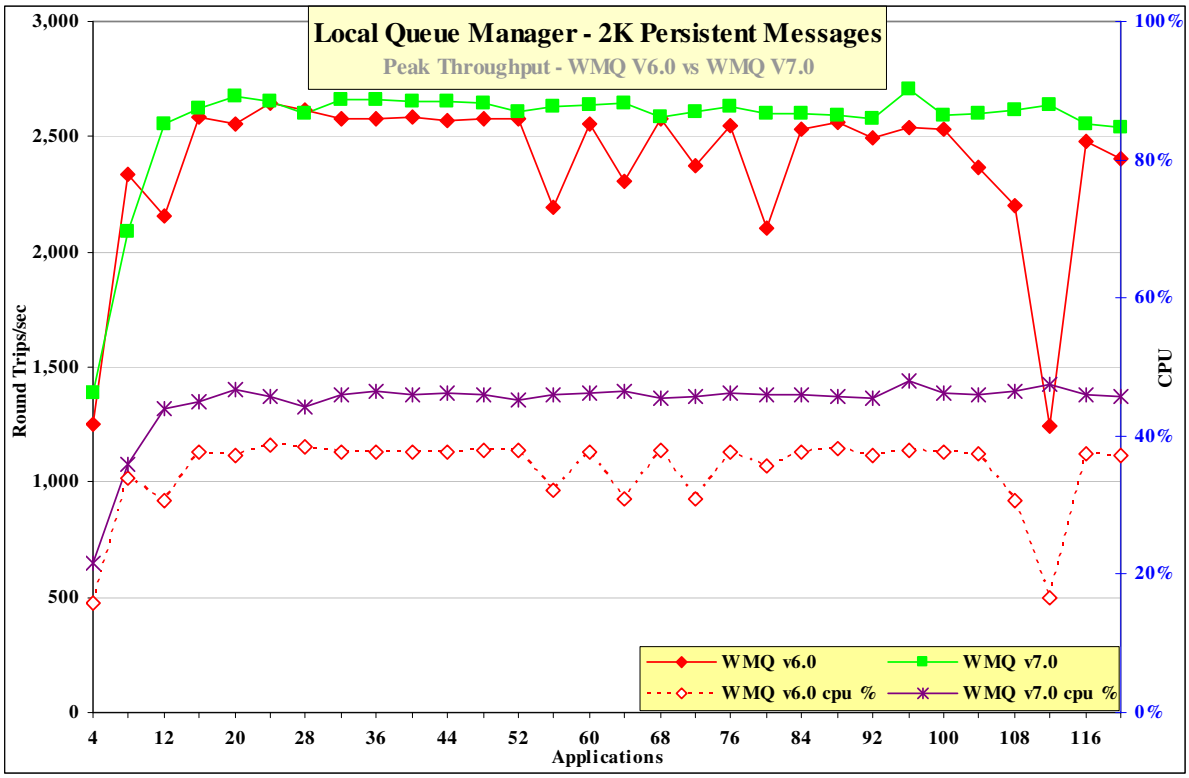


Figure 3 – Performance headline, persistent messages, local queue manager

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

Figure 3 and **Table 2** show that the throughput of persistent messages has improved slightly, about 2-3%, comparing Version 6.0 to Version 7.0.

Test name: **local_pm**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**24** (96)	**2648** (2539)	**0.011** (0.092)	**39%** (38%)
WebSphere MQ V7.0	(24) **96**	(2656) **2708**	(0.011) **0.074**	(46%) **48%**

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 7.

2.2 Client Channels Test Scenario

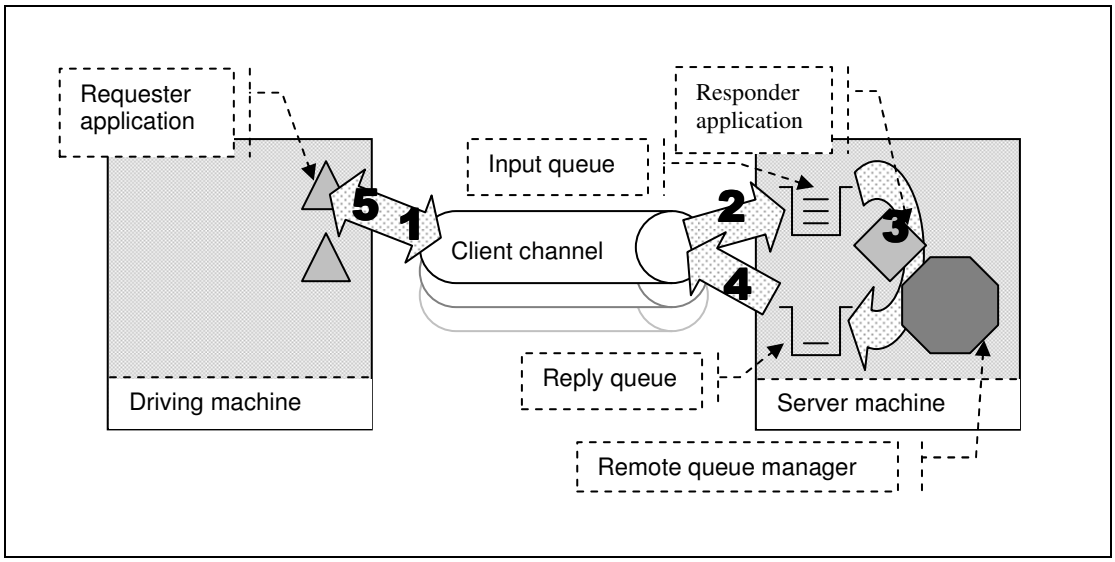


Figure 4 – MQI-client channels into a remote queue manager

1, 2) The Requester 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 Requester application then waits indefinitely for a reply to arrive on the common reply queue.

3) The Responder 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 Requester application gets the reply message (over the client channel), from the common reply queue. The Requester 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.

Application Bindings of the Responder program are 'Shared' and the Client Channel is set to 'MQIBindType = FASTPATH' except in the 'non-trusted' scenario where 'MQIBindType =NORMAL' is used.

Version 7 will multiplex multiple clients over one TCP socket. The version 6 behavior where each client had its own TCP socket can be set by specifying Sharecnv(0) on the client channel definition and is shown in the charts as 'optimized'.

2.2.1 Nonpersistent Messages – Client Channels

Figure 5 , **Figure 5a** 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 V7.0 compared to Version 6.0.

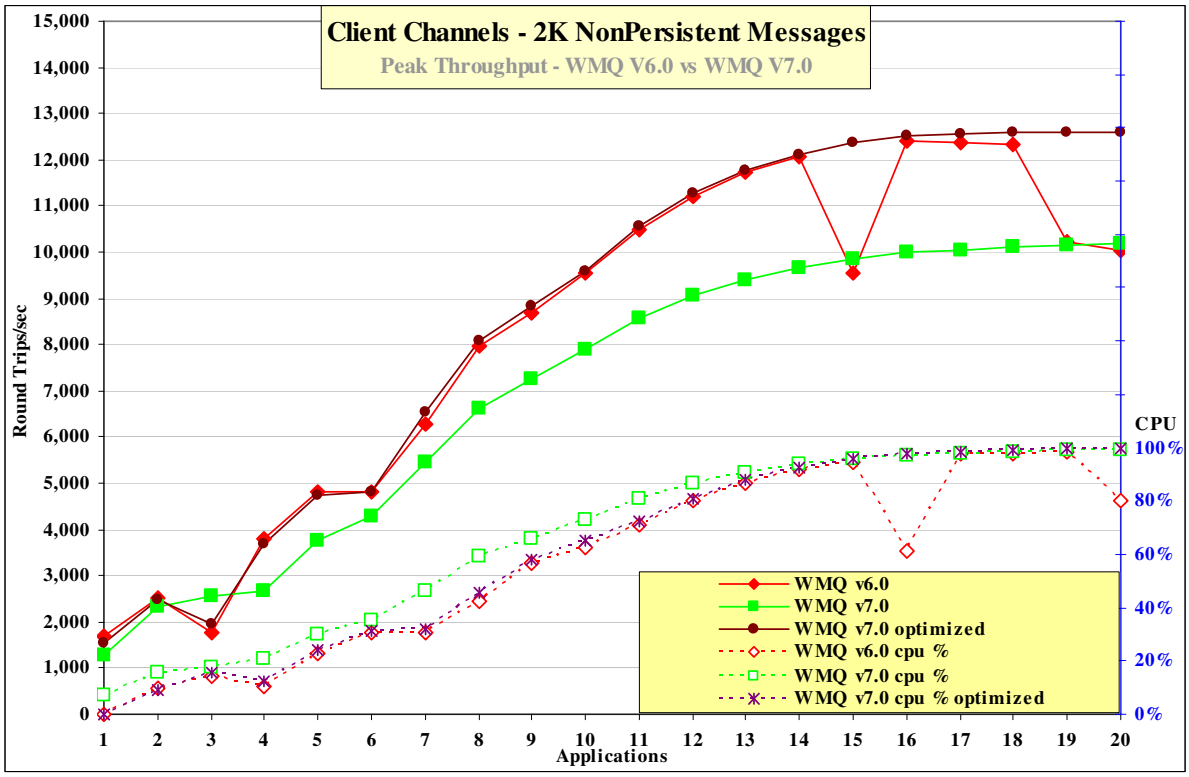


Figure 5 – Performance headline, nonpersistent messages, client channels

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

Figure 5 and **Table 3** show that the throughput of nonpersistent messages in Version 7.0 with optimized setup is equal to Version 6.0, but has degraded by about 19% with the default setup for Version 7.0.

Test name: **clnp**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**16**	**12407**	**0.0015**	**98%**
WebSphere MQ V7.0 (Optimized)	**19**	**12606**	**0.0018**	**100%**
WebSphere MQ V7.0	**20**	**10173**	**0.0023**	**99%**

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.

2.2.2 Nonpersistent Messages – Non Trusted Client Channels

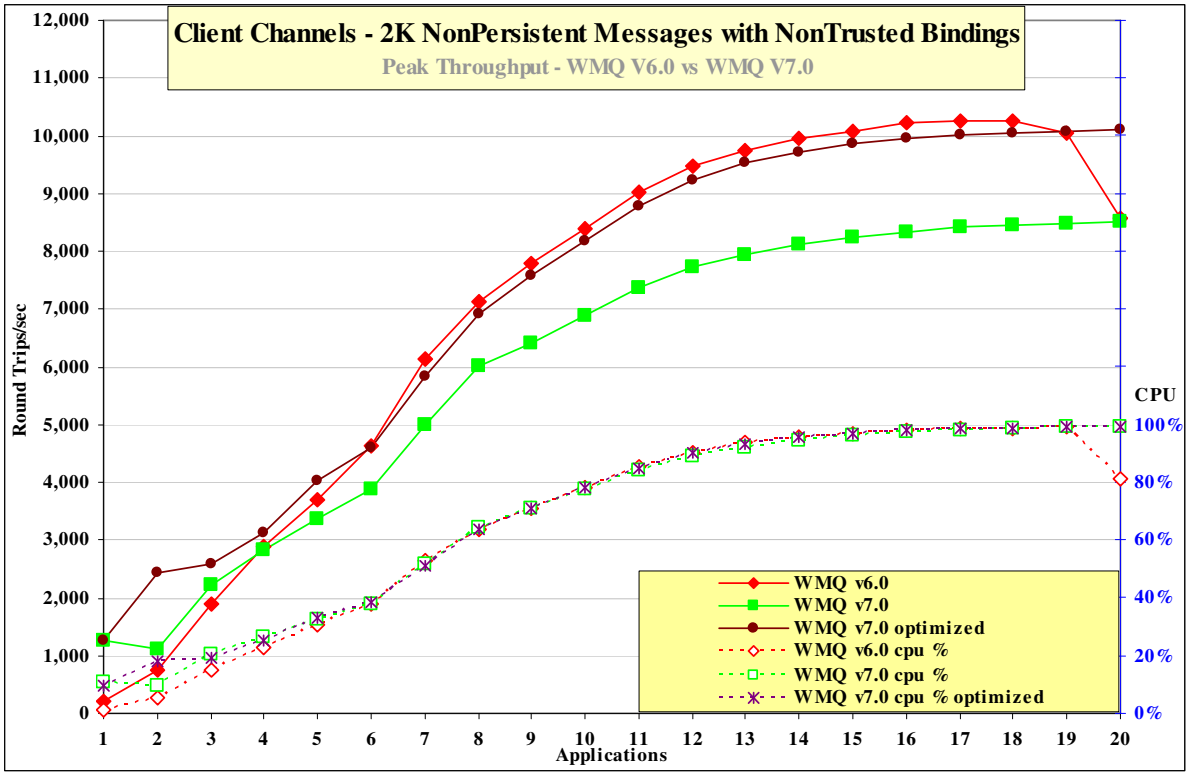


Figure 5a

Test name: **clnp_nt**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**18**	**10244**	**0.0021**	**99%**
WebSphere MQ V7.0 (Optimized)	**20**	**10091**	**0.0023**	**99%**
WebSphere MQ V7.0	(18) **20**	(8448) **8508**	(0.0025 **0.0028**	(99%) **99%**

The throughput of nonpersistent messages when the channel has used the default MQIBINDTYPE=NORMAL has degraded by 1.5% with optimized setup and by 17.0% with default setup when comparing Version 6.0 to Version 7.0.

2.2.3 Persistent Messages – Client Channels

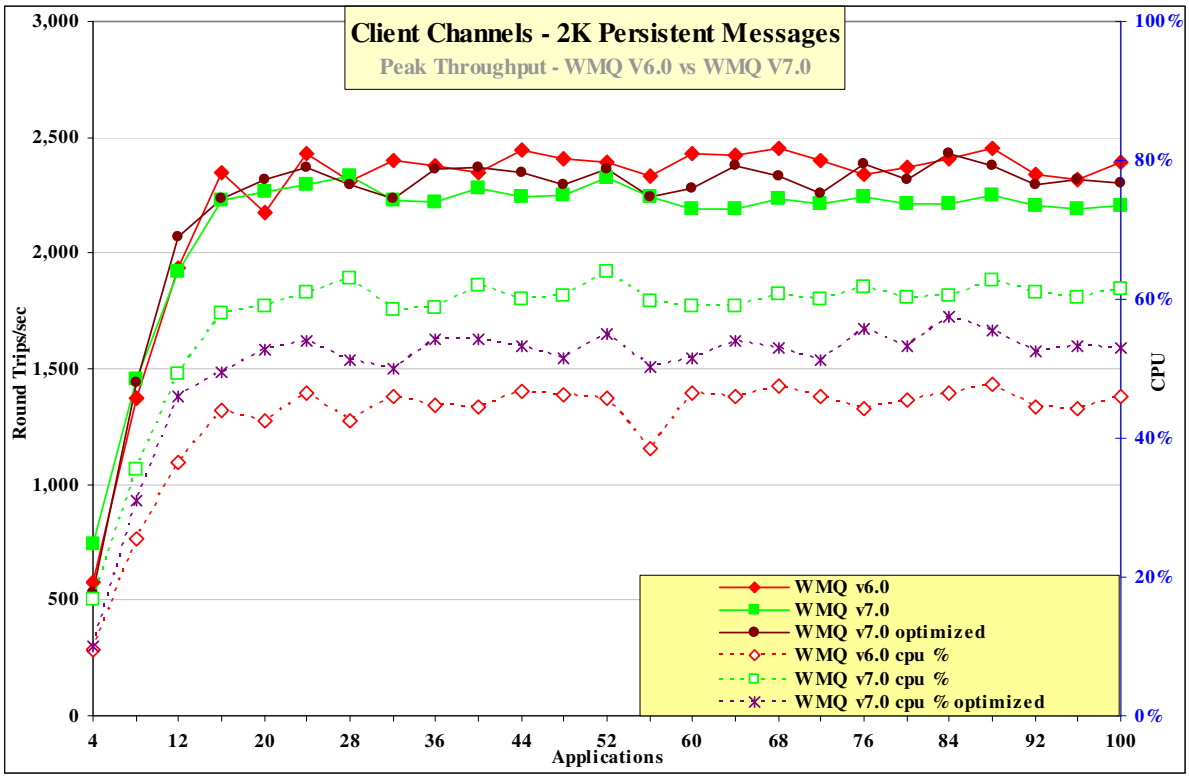


Figure 6 – Performance headline, persistent messages, client channels

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

Figure 6 and **Table 4** that the throughput of persistent messages in Version 7.0 with optimized setup is similar to Version 6.0, but has degraded by about 5% with the default setup for Version 7.0.

Test name: **clpm**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**68**	**2453**	**0.033**	**48%**
WebSphere MQ V7.0 (Optimized)	**108**	**2454**	**0.053**	**58%**
WebSphere MQ V7.0	**28** (68)	**2330** (2232)	**0.014** (0.036)	**63%** (61%)

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 7.

2.2.4 Client Channels

For the following client channel measurements, the message 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.

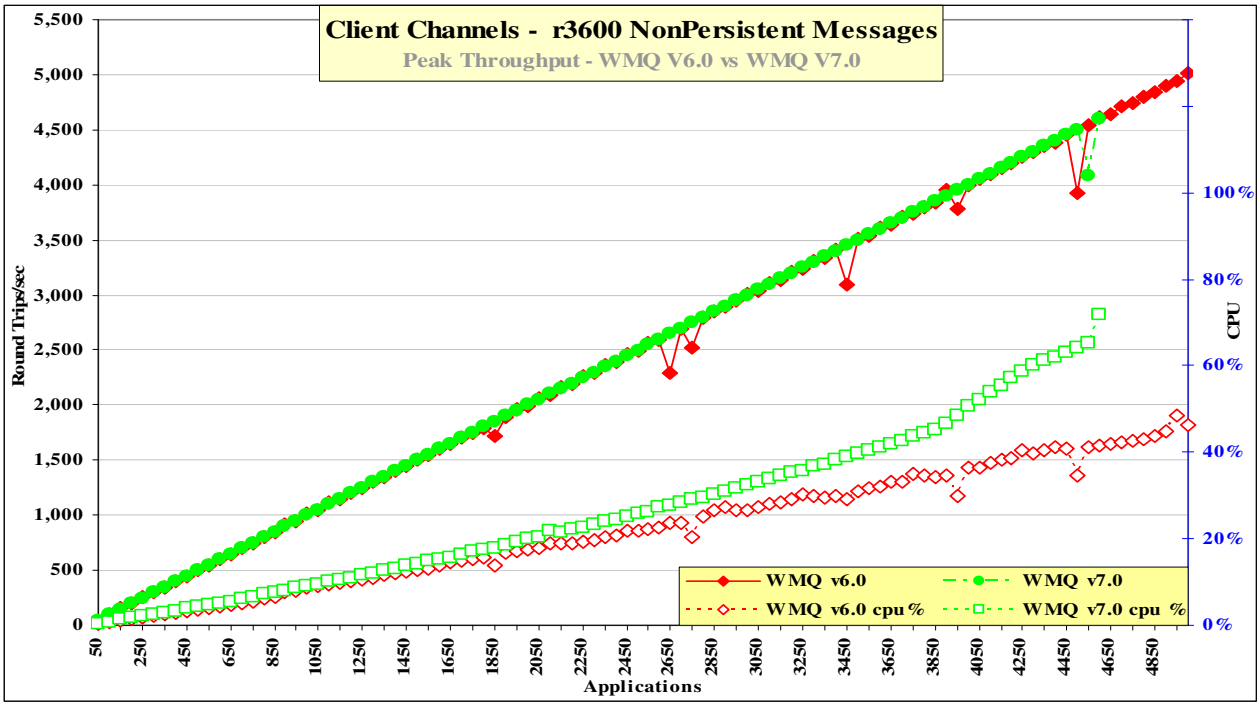


Figure 7 – 1 round trip per driving application per second, client channels, nonpersistent messages

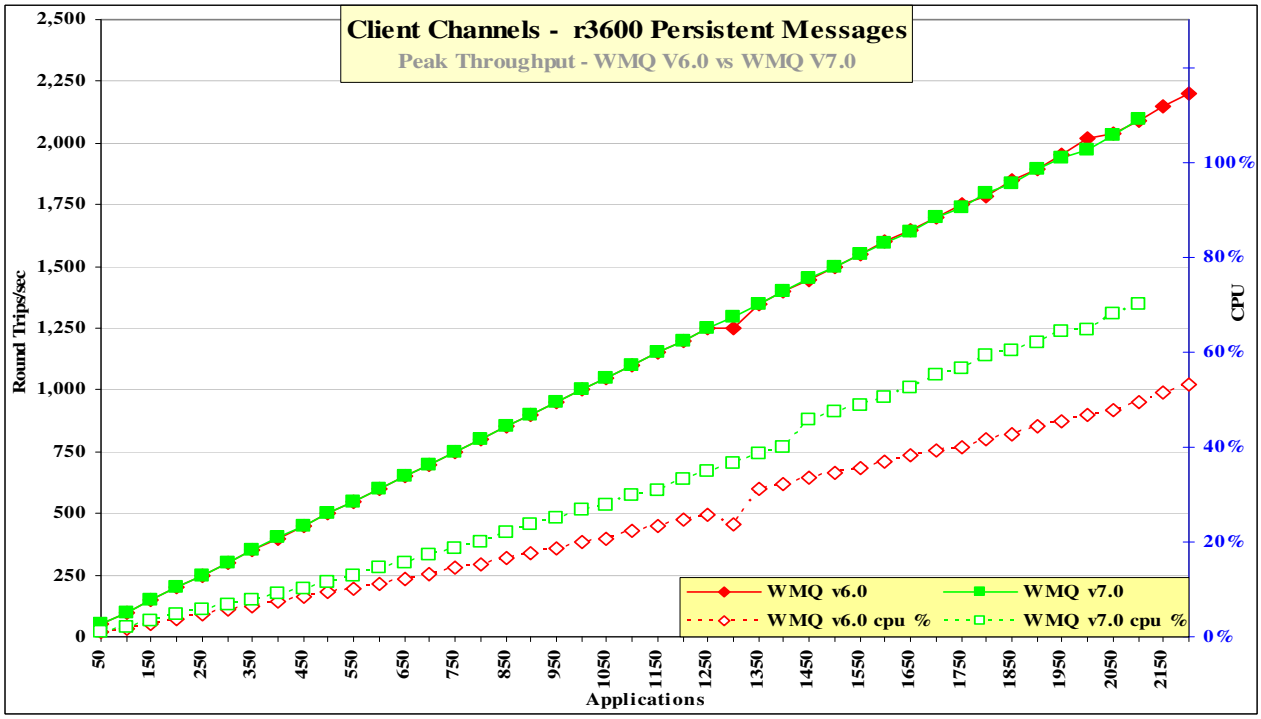


Figure 8 – 1 round trip per driving application per second, client channels, persistent messages

Figure 7,

Figure 8 and **Table 5** shows that maximum throughput levels for WebSphere MQ V7.0 are about 10% less for nonpersistent messaging and about 9% less for persistent messaging when compare with WebSphere MQ V6.0.

Test name:	**Apps**	**Rate/app/hr**	**Round Trips/sec**	**Response Time**	**CPU**
clnp_r3600 (WebSphereMQ v6.0)	**4500** (5000)	3600	**4499** (5006)	**0.005** (0.0128)	**64%** (46%)
clpm_r3600 (WebSphereMQ v6.0)	**2100** (2300)	3600	**2097** (2284)	**0.4732** (1.1677)	**70%** (54%)

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 V7.0 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 6.

2.3 Distributed Queuing Test Scenario

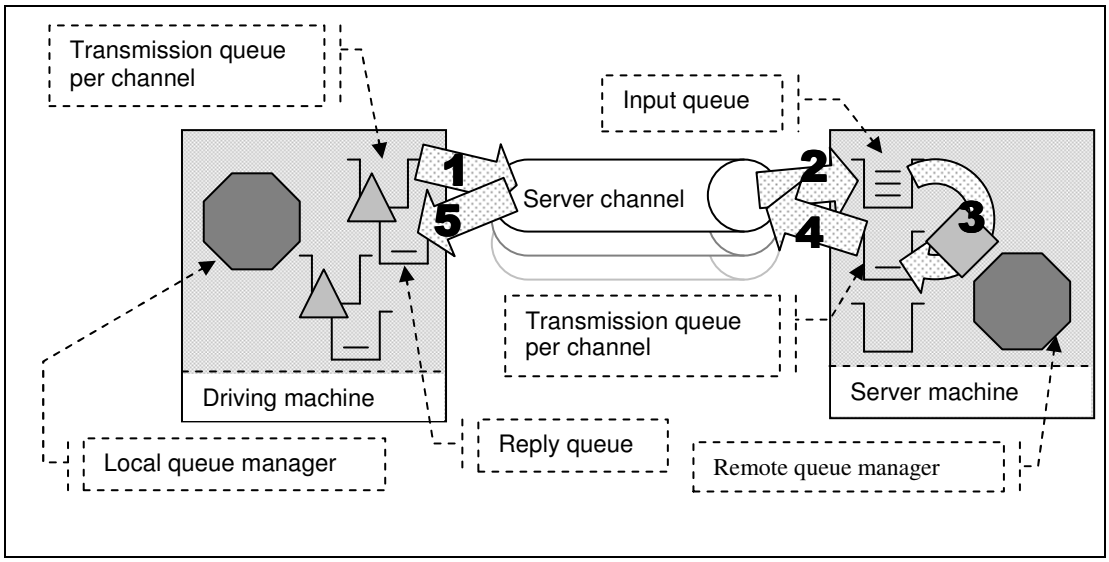


Figure 9 – Server channels between two queue managers

1) The Requester 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 Requester 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 Responder 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 Requester application gets a reply from a local queue. The Requester 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.

Application Bindings of the Responder program are 'Shared' , the Requester program is normally 'Trusted' , and the channels specified as 'MQIBindType = FASTPATH' except in the 'non-trusted' scenario where both programs use 'shared' bindings and the channels are specified as 'MQIBindType = NORMAL'.

2.3.1 Nonpersistent Messages – Server Channels

Figure 10 , **Figure 10a** , and **Figure 11** show the nonpersistent and persistent message throughput achieved using an increasing number of driving applications in the distributed queuing scenario (see **Figure 9** on the previous page), and WebSphere MQ V6.0 compared to Version 7.

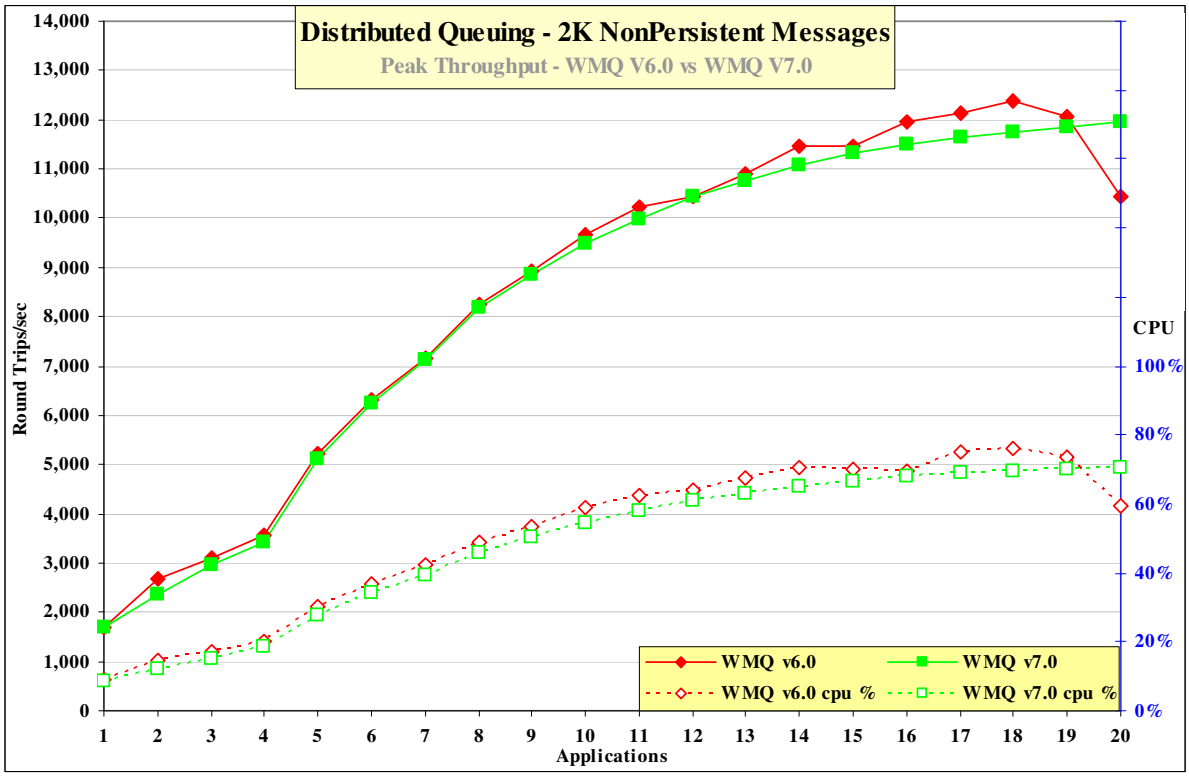


Figure 10 – Performance headline, nonpersistent messages, server channels

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

Figure 10 and **Table 6** show that the throughput of nonpersistent messages in Version 7 is similar to Version 6.

Test name: **dqnp**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**18**	**12366**	**0.0018**	**76%**
WebSphere MQ V7.0	(18) **20**	(11755) **11942**	(0.0019) **0.0020**	(70%) **71%**

Table 6 – 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 7.

2.3.2 Non-Persistent Non-Trusted – Server Channels

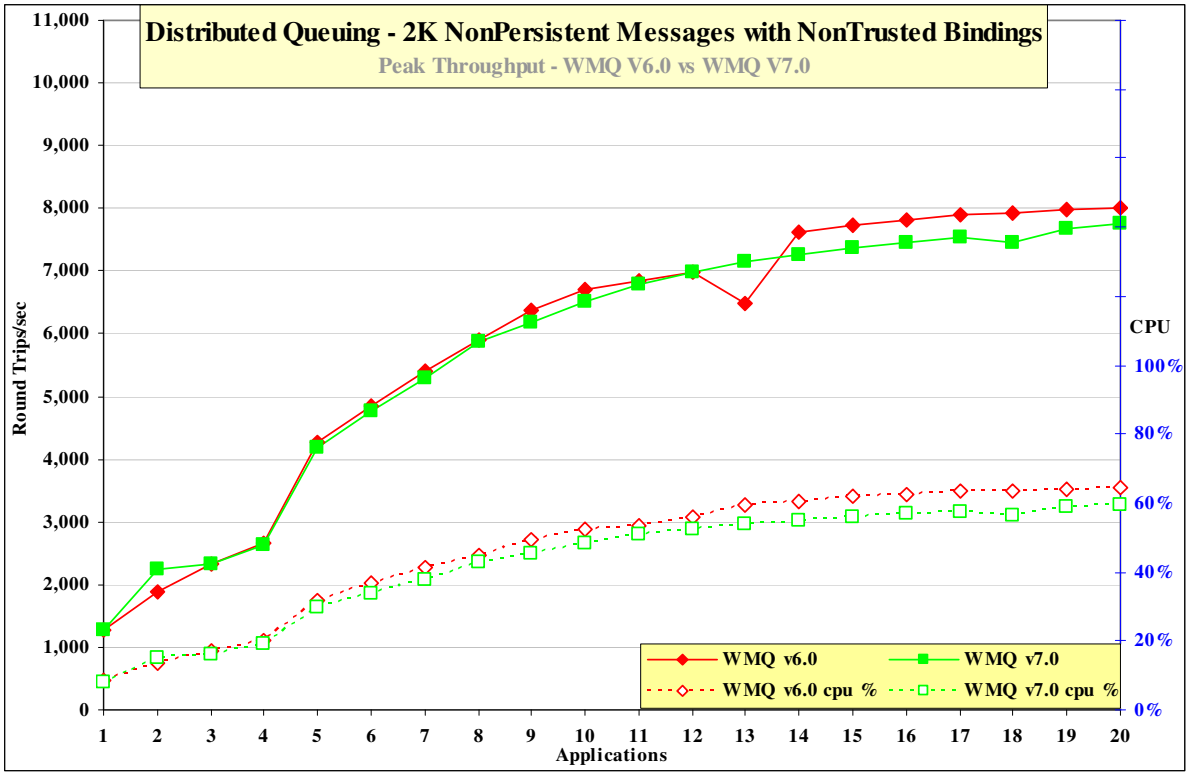


Figure 10a

The throughput of nonpersistent messages in Version 7 is similar to Version 6 when the channel is using MQIBINDTYPE=NORMAL.

Test name: **dqnp_nt**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**20**	**8010**	**0.0029**	**64%**
WebSphere MQ V7.0	**20**	**7756**	**0.0033**	**59%**

2.3.3 Persistent Messages – Server Channels

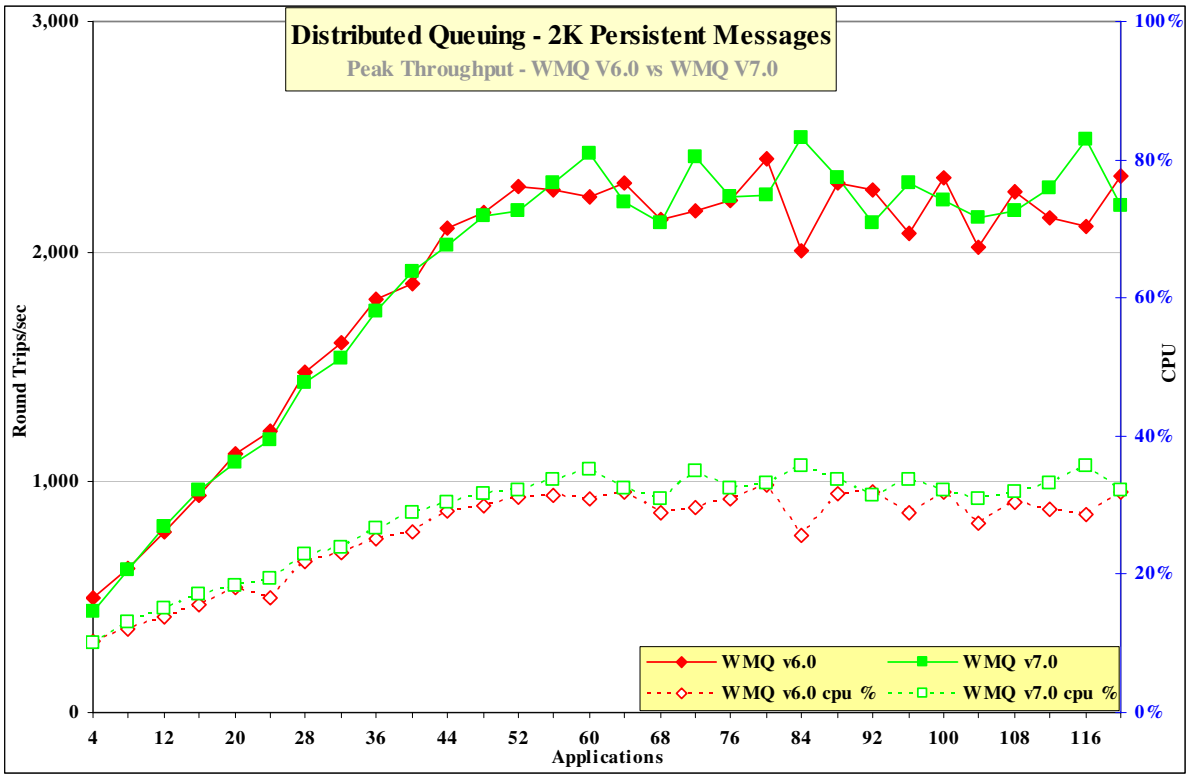


Figure 11 – Performance headline, persistent messages, server channels

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

Figure 11 and **Table 7** show that the throughput of persistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqpm**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**80**	**2405**	**0.0408**	**33%**
WebSphere MQ V7.0	**84**	**2496**	**0.0416**	**36%**

Table 7 – 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 7.

2.3.4 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.

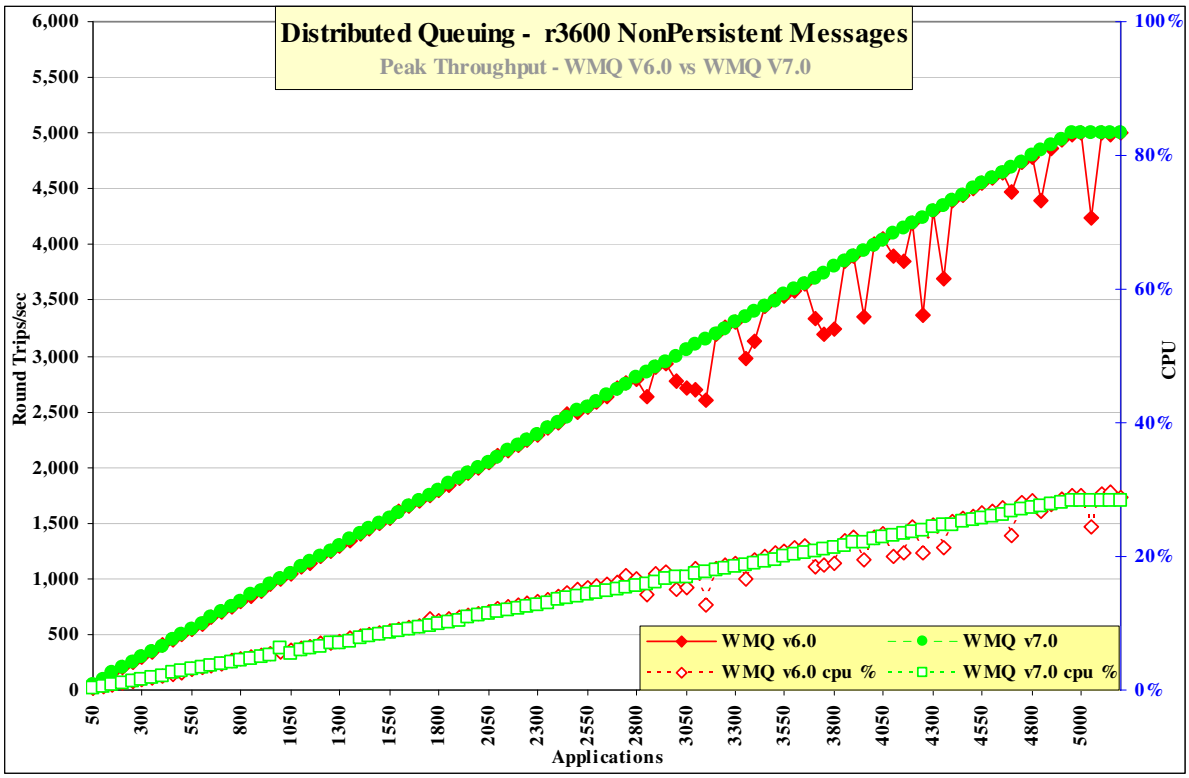


Figure 12 – 1 round trip per driving application per second, server channel, nonpersistent messages

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

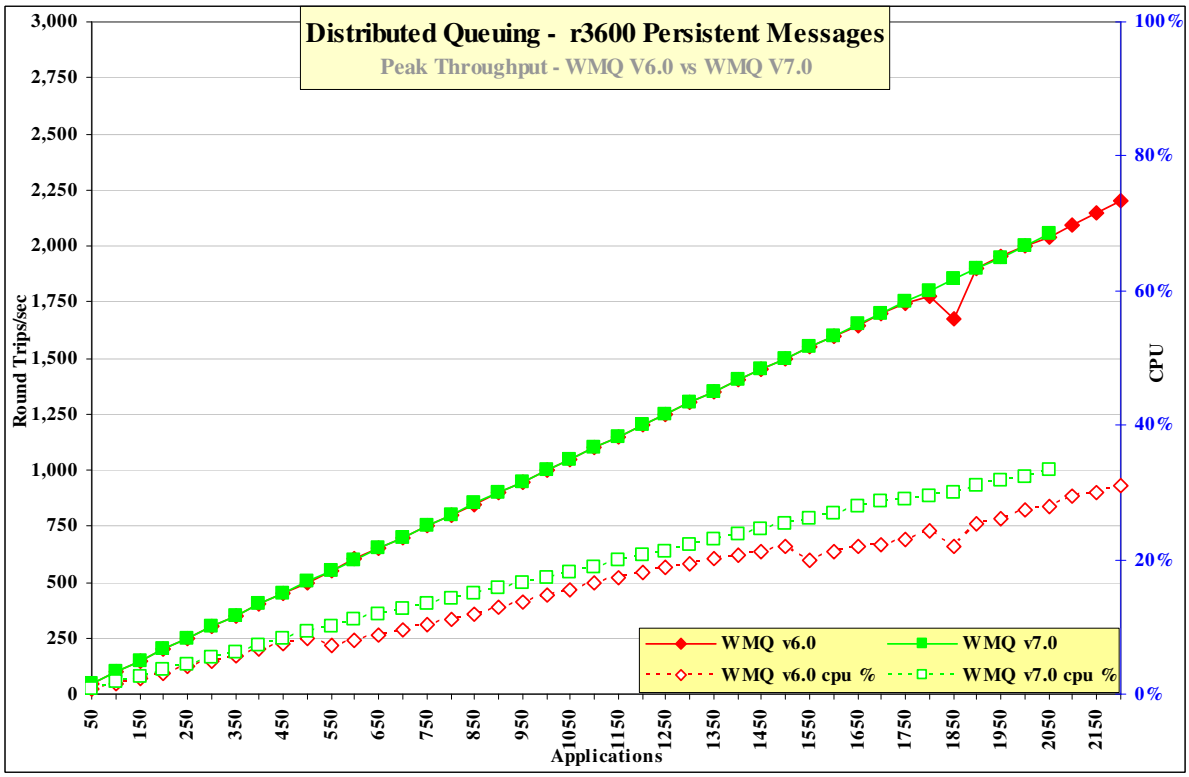


Figure 13 – 1 round trip per driving application per second, server channel, persistent messages

Figure 12, **Figure 13** and **Table 8** show that maximum throughput levels for WebSphere MQ V7.0 are similar for nonpersistent messaging and about 6.8% less for persistent messaging when compare with WebSphere MQ V6.0.

Test name:	Apps	Rate/app/hr	Round Trips/sec	Response Time	CPU
dqnp_r3600 (WebSphereMQ v6.0)	**5000** (5000)	3600	**4998** (4999)	**0.0018** (0.0027)	**28%** (29%)
dqpm_r3600 (WebSphereMQ v6.0)	**2050** (2200)	3600	**2053** (2198)	**0.0719** (0.1860)	**33%** (31%)

Table 8 – 1 round trip per driving application per second, client channels

Note: The large bold numbers in the table above show the WebSphere MQ V7.0 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 6.

3 Large Messages

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

3.1.1 50bytes to 32KB

Figure 14 and **Figure 15** show that the response times for MQPUT/GET for both nonpersistent and persistent message sizes between 50bytes and 32KB has remained about the same for WebSphere MQ Version 7.

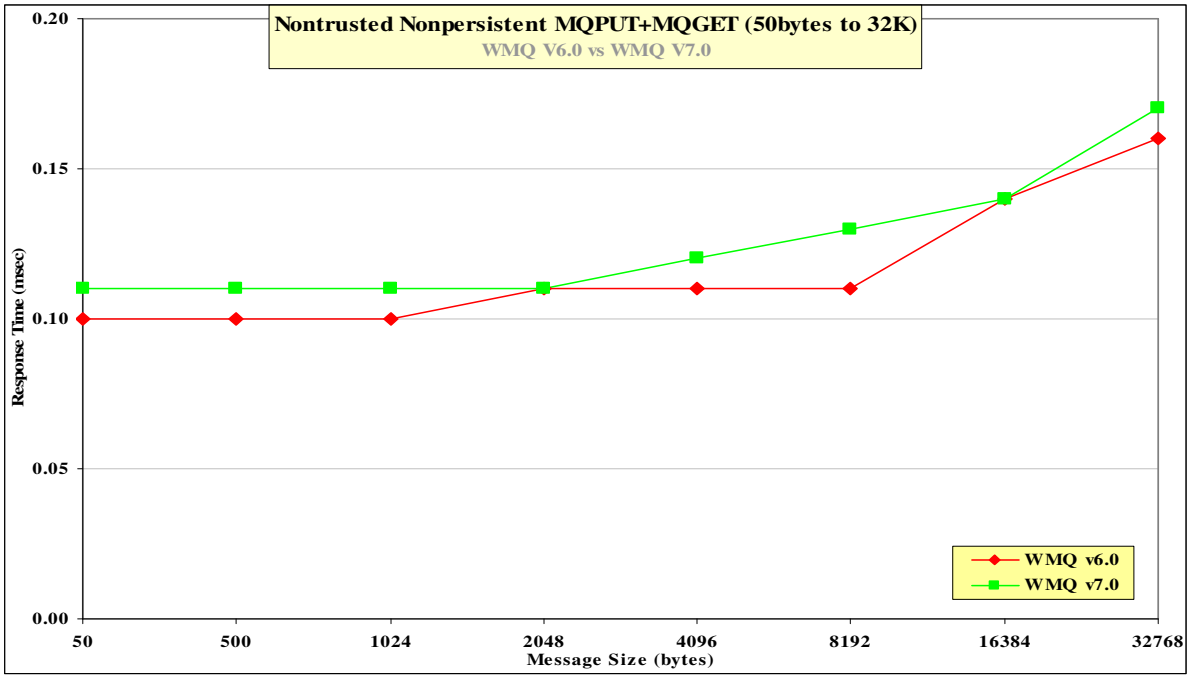


Figure 14 –The effect of nonpersistent message size on MQI response time (50byte - 32K)

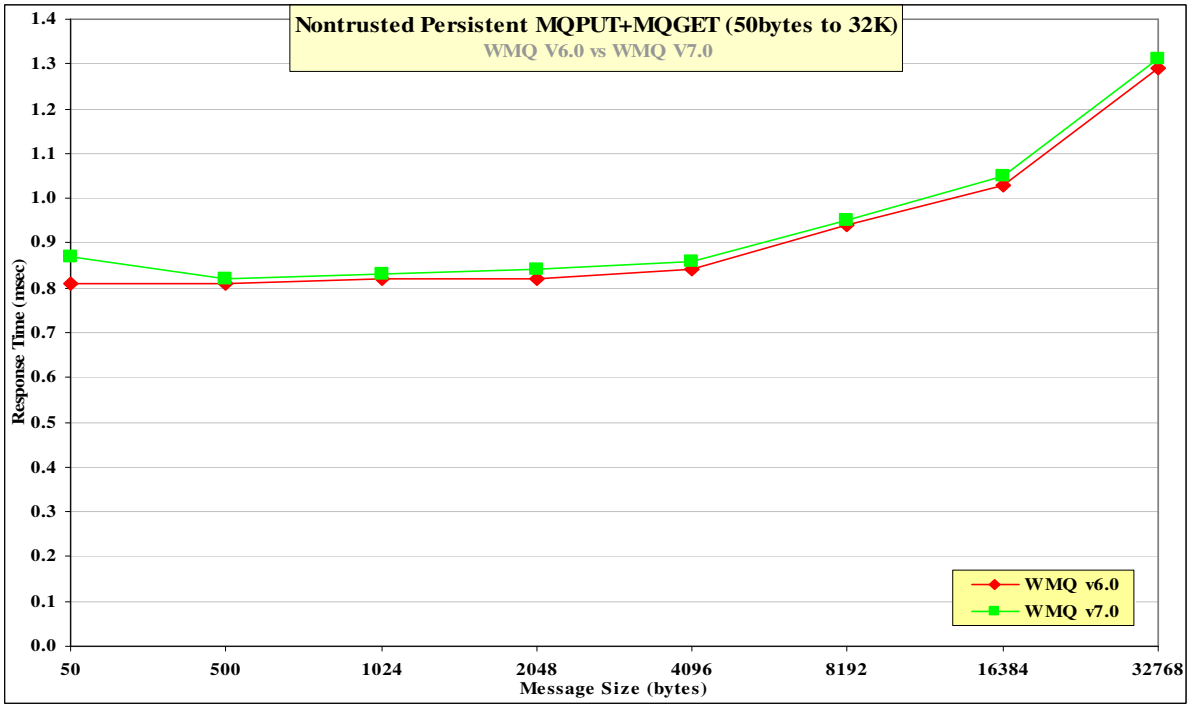


Figure 15 –The effect of persistent message size on MQI response time (50byte - 32K)

3.1.2 32KB to 2MB

Figure 16 and **Figure 17** show show that the response times for MQPUT/GET for both nonpersistent and persistent message sizes between 32KB and 2MB has remained about the same for WebSphere MQ Version 7.

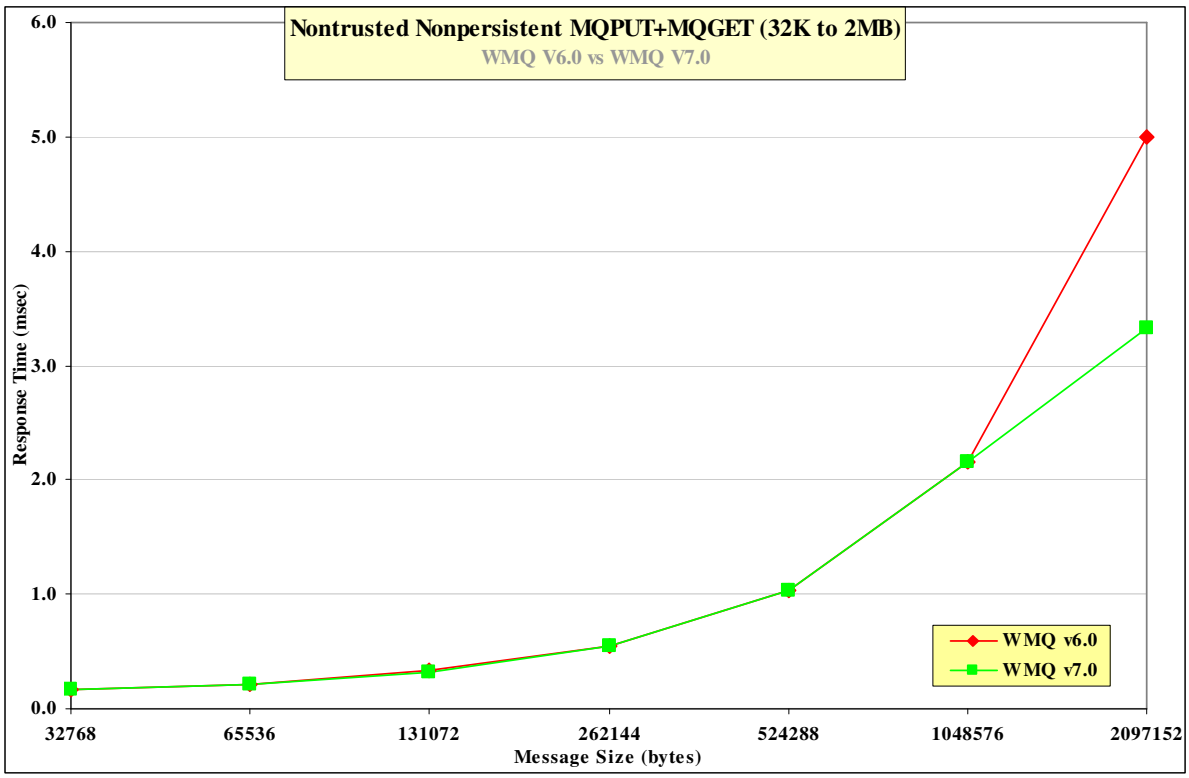


Figure 16 –The effect of nonpersistent message size on MQI response time (32K – 2MB)

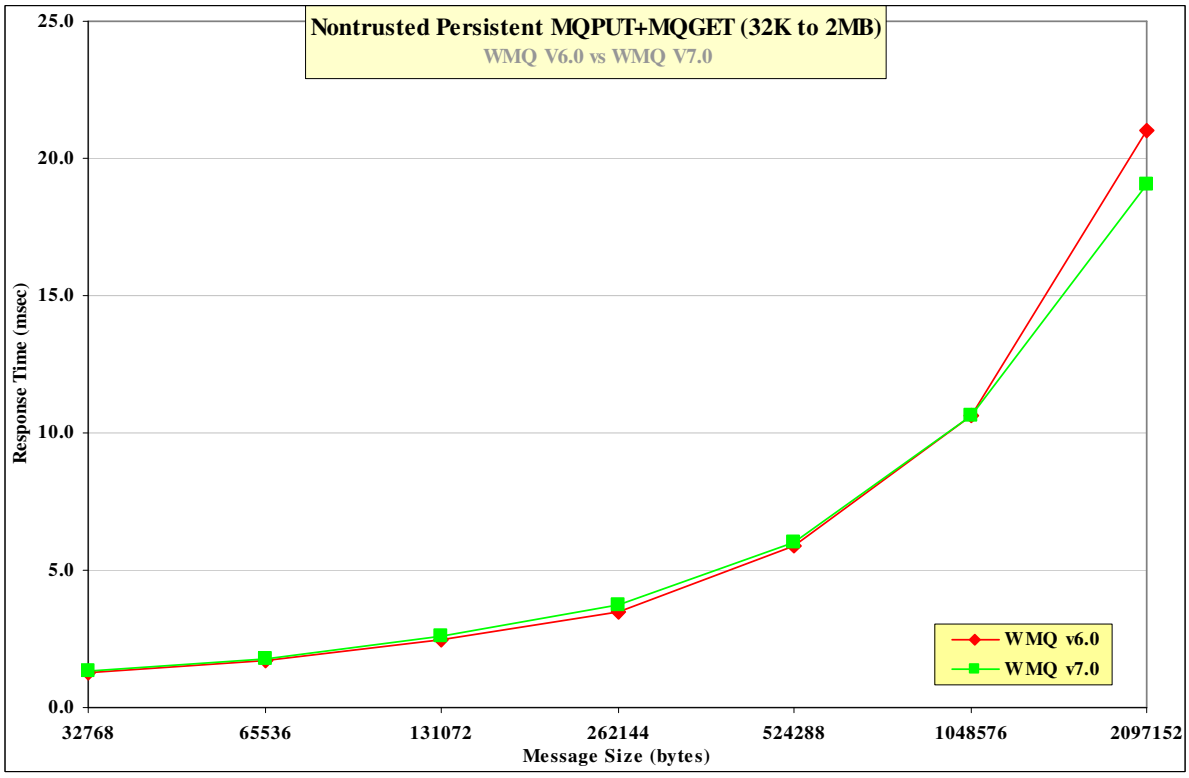


Figure 17 –The effect of persistent message size on MQI response time (32K – 2MB)

3.1.3 2MB to 8MB

Figure 18 and **Figure 19** show that the response time for MQPUT/GET pairs has improved noticeably for nonpersistent and persistent message sizes between 2MB and 8MB.

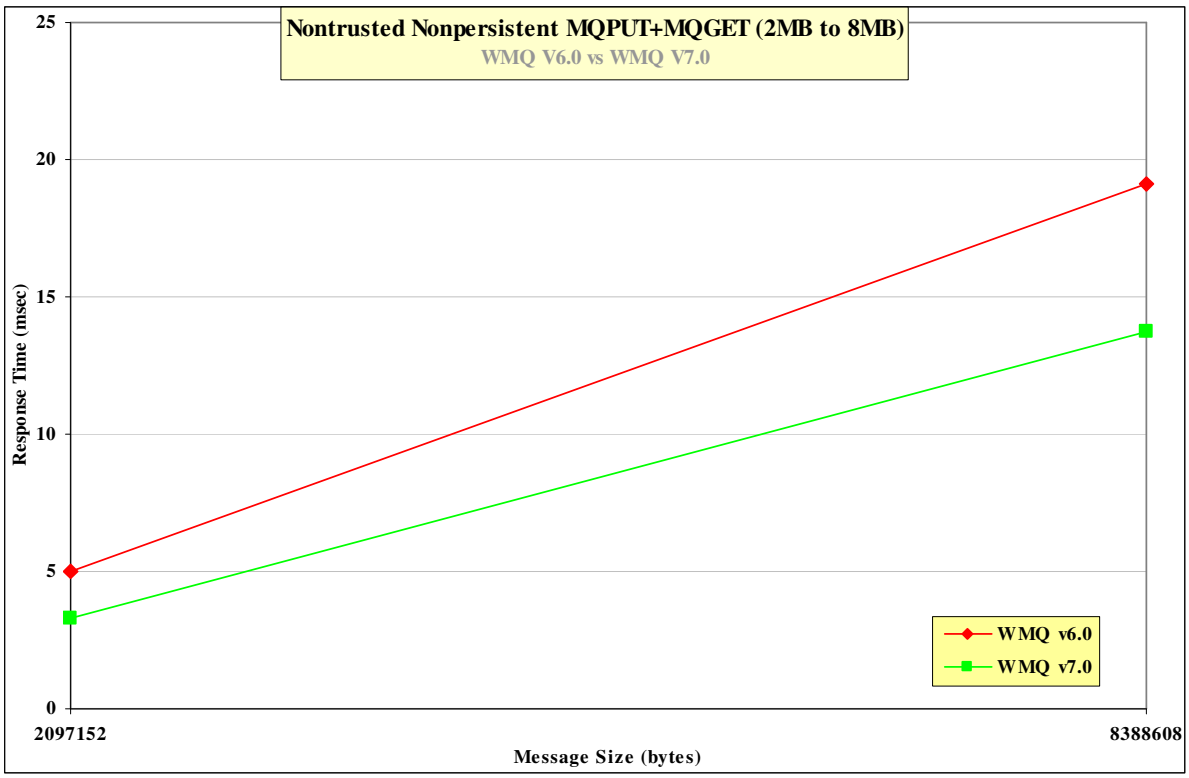


Figure 18 –The effect of nonpersistent message size on MQI response time (2MB – 8MB)

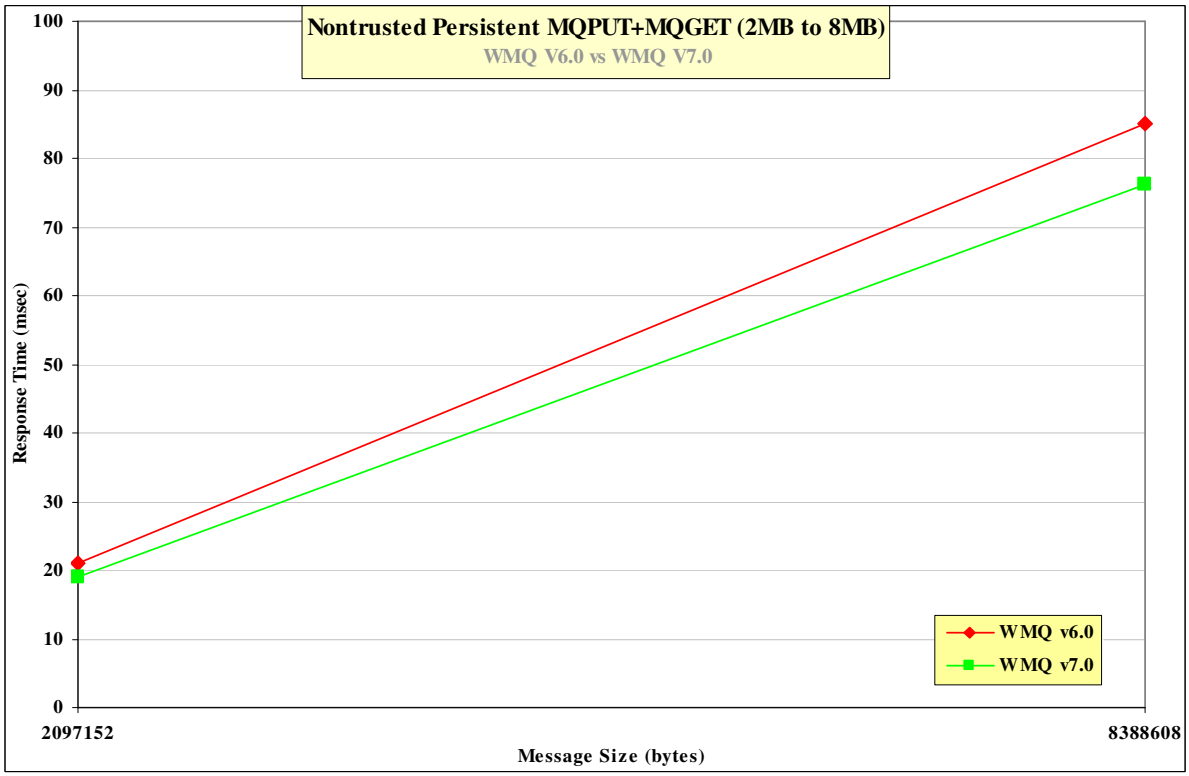


Figure 19 –The effect of persistent message size on MQI response time (2MB – 8MB)

3.2 20K Messages

3.2.1 Local Queue Manager

Figure 20 and **Figure 21** 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

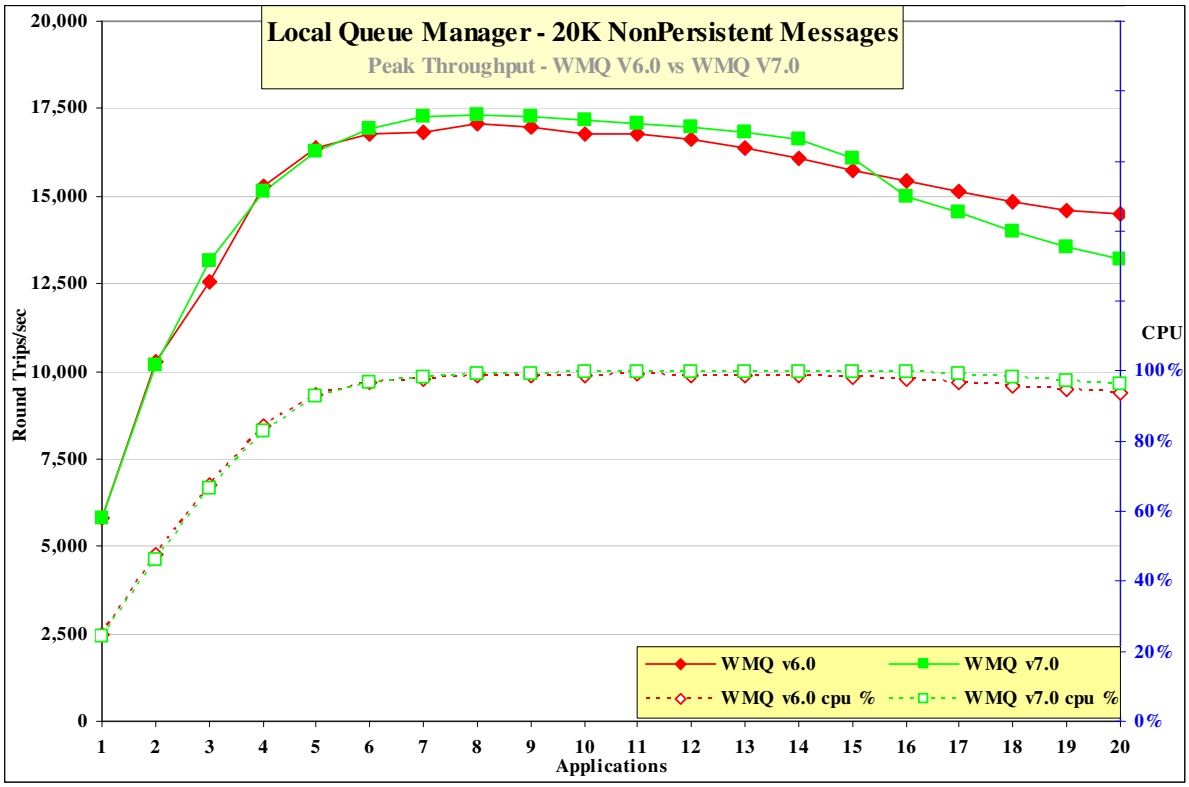


Figure 20 – 20K nonpersistent messages, local queue manager

Figure 20 and **Table 9** show that the throughput of nonpersistent messages is similar comparing Version 7.0 to Version 6.0.

Test name: **local_np_20K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**8**	**17078**	**0.0006**	**99%**
WebSphere MQ V7.0	**8**	**17318**	**0.0006**	**99%**

Table 9 – 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.

3.2.1.2 Persistent Messages

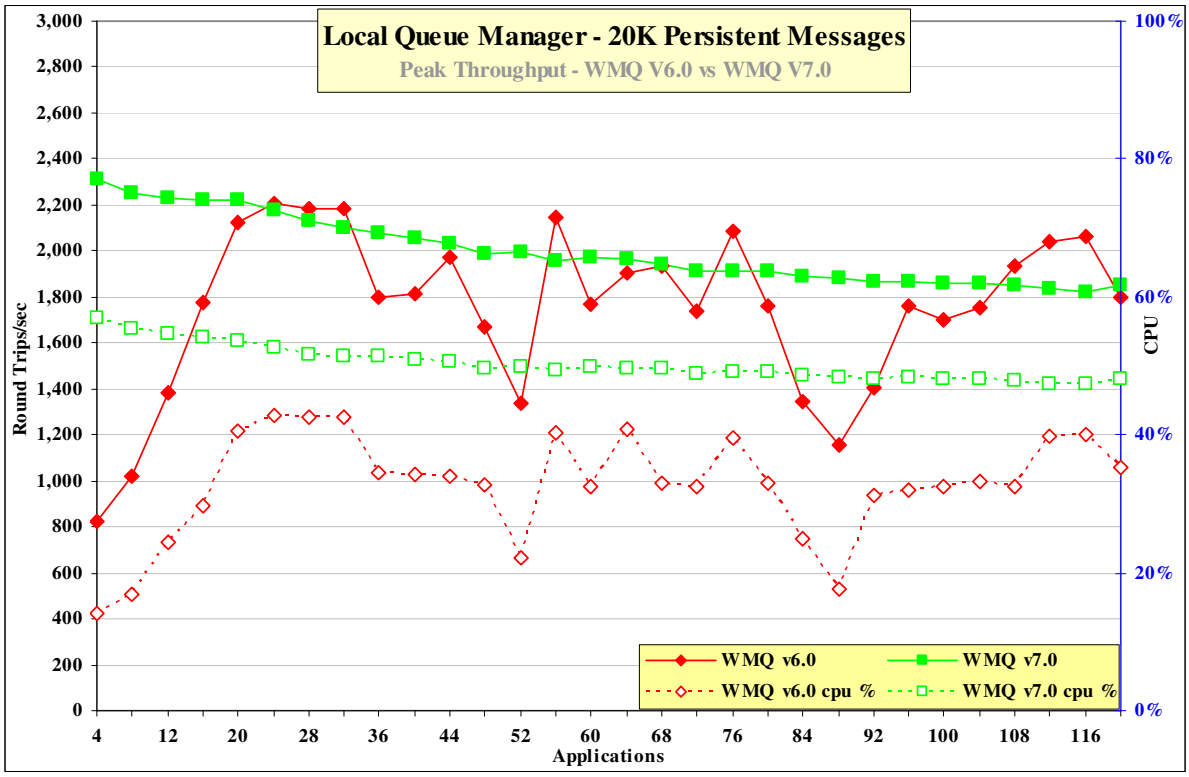


Figure 21 – 20K persistent messages, local queue manager

Figure 21 and **Table 10** show that the throughput of persistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **local_pm_20K**	Apps	Round Trips/sec	Response Time	CPU
WebSphere MQ V6.0	**24**	**2203**	**0.0134**	**43%**
WebSphere MQ V7.0	**4** (24)	**2314** (2094)	**0.0153** (0.0145)	**57%** (42%)

Table 10 – 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 7.

3.2.2 Client Channels

Figure 22 and **Figure 23** 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

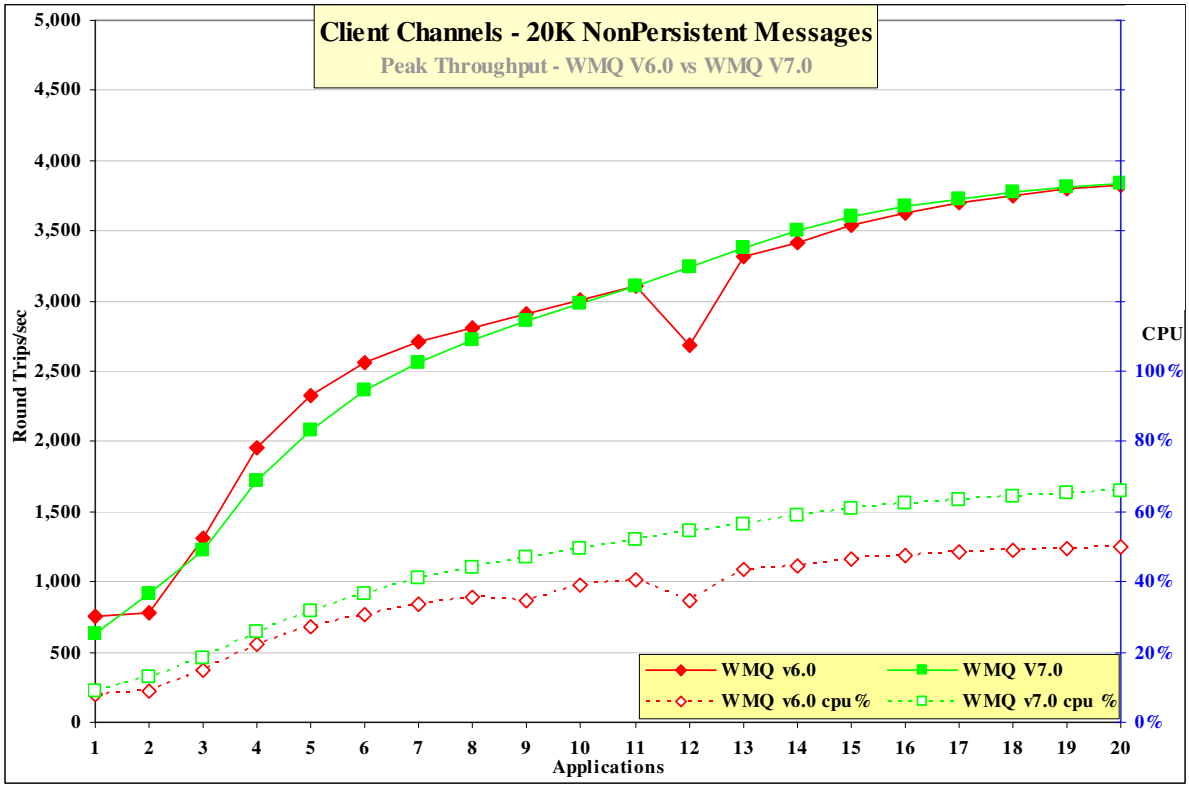


Figure 22 – 20K nonpersistent messages, client channels

Figure 22 and **Table 11** show that that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **clnp_20K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**20**	**3823**	**0.0062**	**50%**
WebSphere MQ V7.0	**20**	**3840**	**0.0062**	**66%**

Table 11 – 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.

3.2.2.2 Persistent Messages

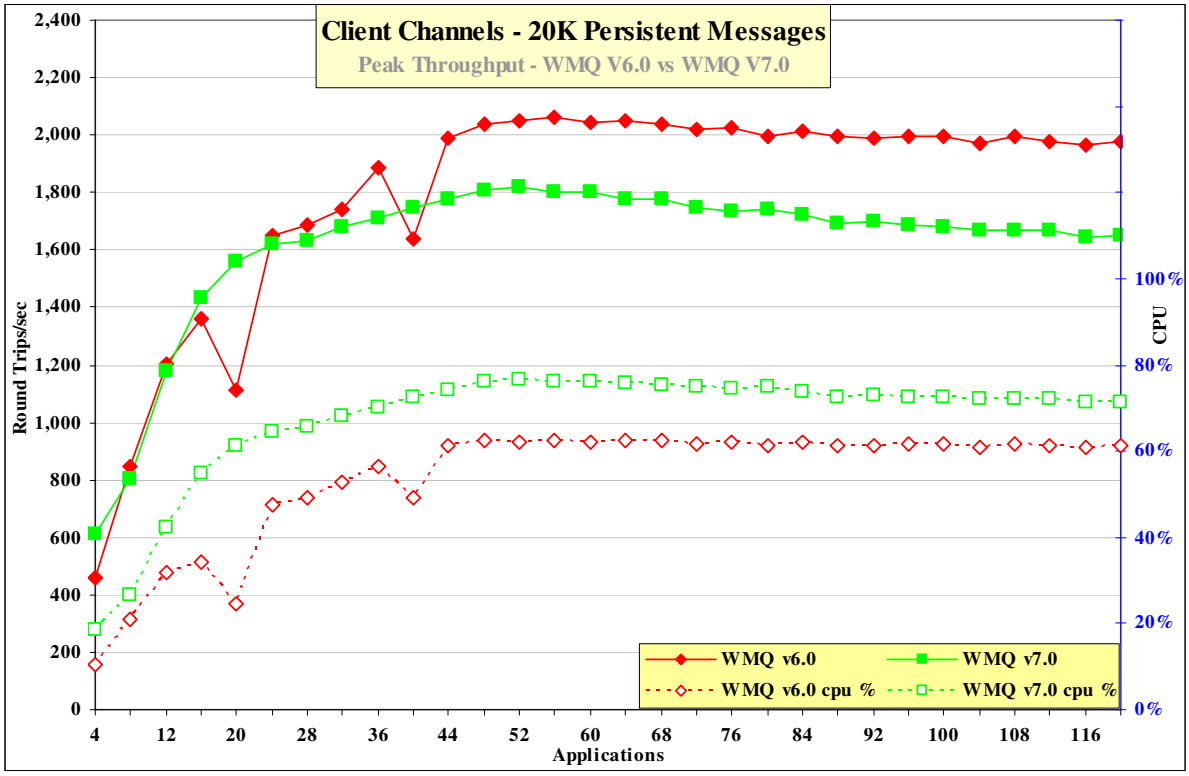


Figure 23 – 20K persistent messages, client channels

Figure 23 and **Table 12** show that the throughput of persistent messages has degraded by 11.7% when comparing Version 6.0 to Version 7.0.

Test name: **clpm_20K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(52) **56**	(2051) **2063**	(0.0287) **0.0341**	(62%) **63%**
WebSphere MQ V7.0	**52** (56)	**1822** (1801)	**0.0331** (0.0357)	**77%** (76%)

Table 12 – 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 7.

3.2.3 Distributed Queuing

Figure 24 and **Figure 25** 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

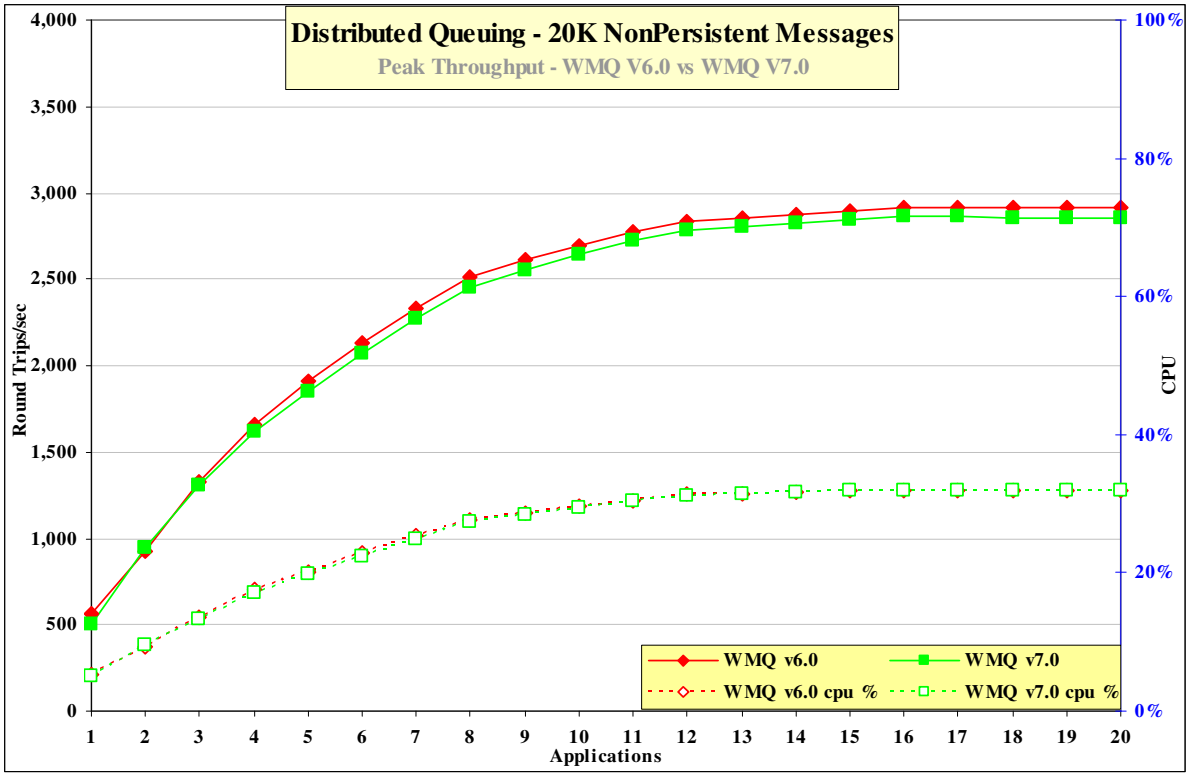


Figure 24 – 20K nonpersistent messages, distributed queuing

Figure 24 and **Table 13** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqnp_20K**	Apps	Round Trips/sec	Response Time	CPU
WebSphere MQ V6.0	(17) **20**	(2910) **2917**	(0.0071) **0.0082**	(32%) **32%**
WebSphere MQ V7.0	**17** (20)	**2861** (2855)	**0.0072** (0.0085)	**32%** (32%)

Table 13 – 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 7.

3.2.3.2 Persistent Messages

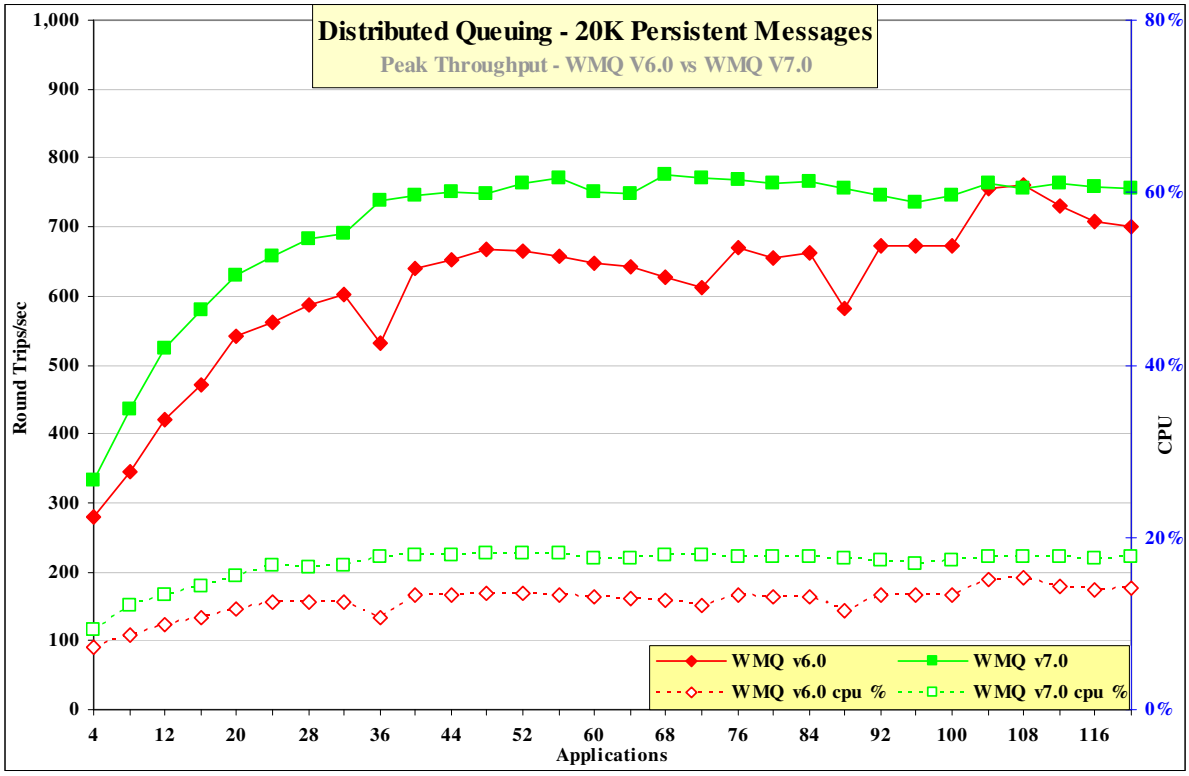


Figure 25 – 20K persistent messages, distributed queuing

Figure 25 and **Table 14** show that the throughput of persistent messages has improved by 10-20% when comparing Version 6.0 to Version 7.0.

Test name: **dqpm_20K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(68) **112**	(627) **761**	(0.1233) **0.1725**	(13%) **14%**
WebSphere MQ V7.0	**68** (112)	**775** (734)	**0.1023** (0.1734)	**18%** (19%)

Table 14 – 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 7.

3.3 200K Messages

3.3.1 Local Queue Manager

Figure 26 and **Figure 27** 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

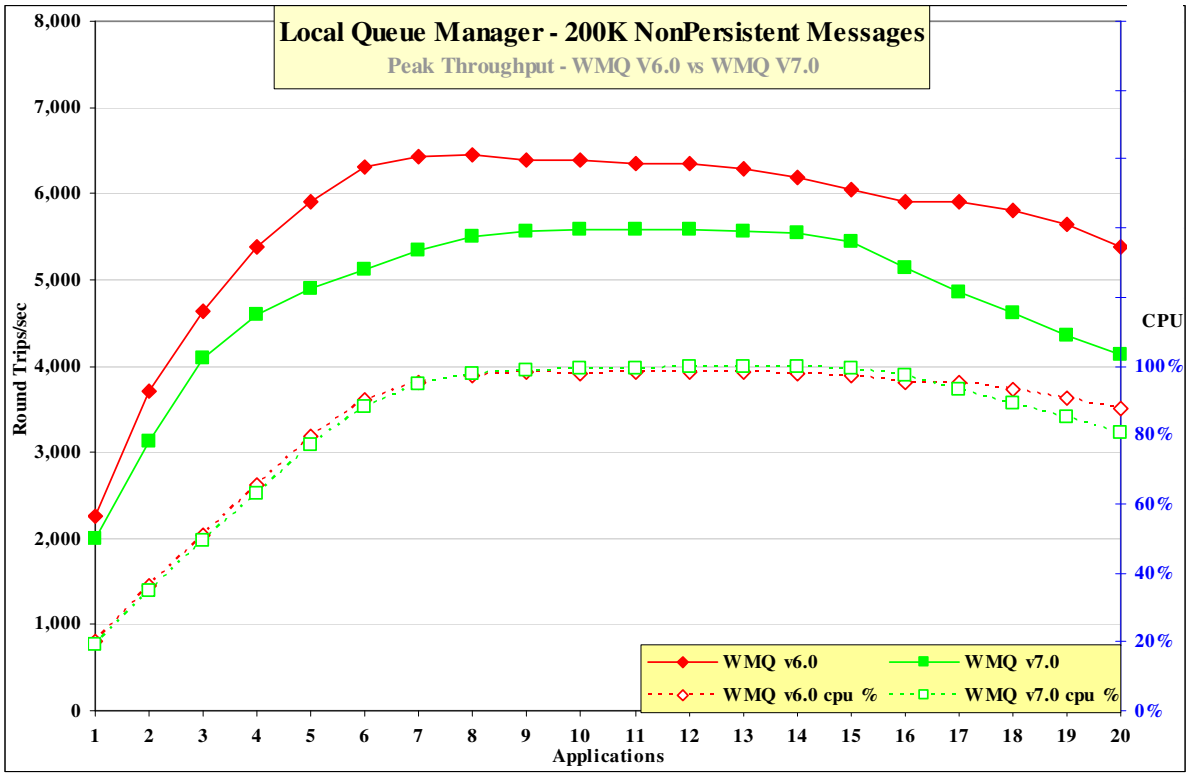


Figure 26 – 200K nonpersistent messages, local queue manager

Figure 26 and **Table 15** show that the throughput of nonpersistent messages has degraded by about 13-14% when comparing Version 6.0 to Version 7.0.

Test name: **local_np_200K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**8** (11)	**6446** (6357)	**0.0015** (0.0021)	**97%** (98%)
WebSphere MQ V7.0	(8) **11**	(5500) **5588**	(0.0017) **0.0024**	(98%) **99%**

Table 15 – 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 7

3.3.1.2 Persistent Messages

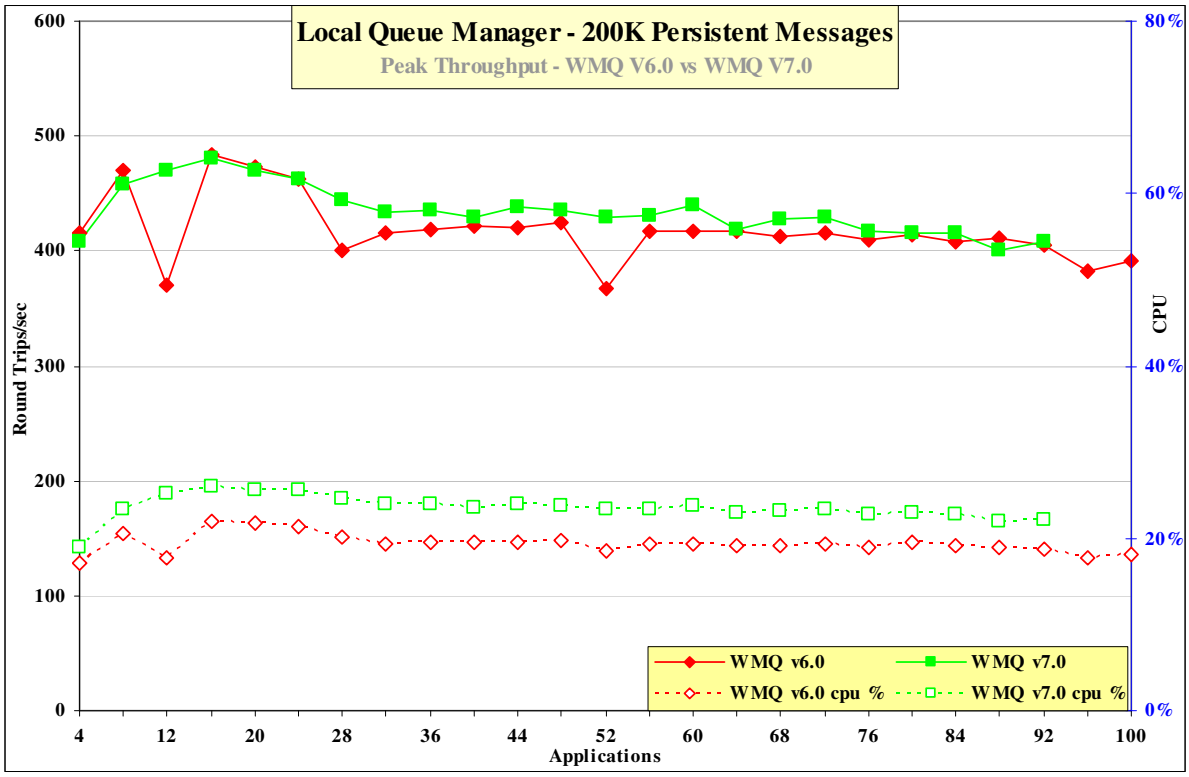


Figure 27 – 200K persistent messages, local queue manager

Figure 27 and **Table 16** show that the throughput of persistent messages is similar comparing Version 6.0 to Version 7.0.

Test name: **local_pm_200K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**16**	**483**	**0.0657**	**22%**
WebSphere MQ V7.0	**16**	**480**	**0.0426**	**26%**

Table 16 – 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.

3.3.2 Client Channel

Figure 28 and **Figure 29** 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

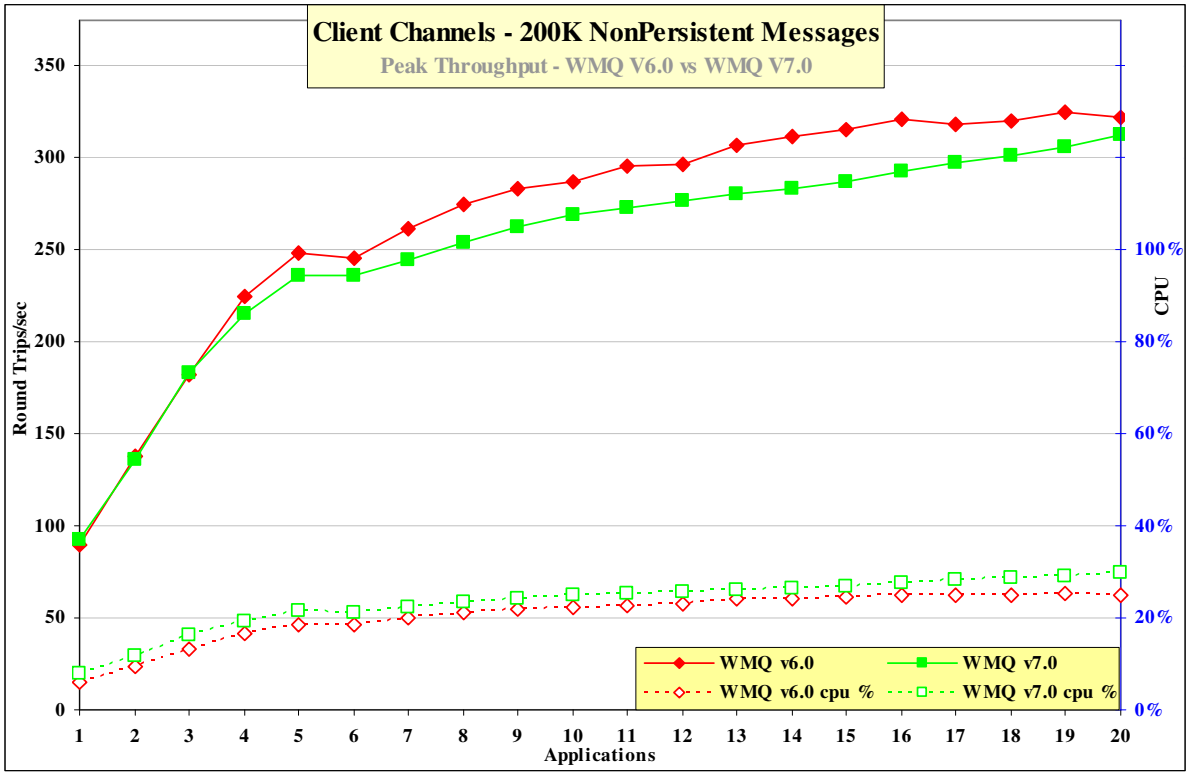


Figure 28 – 200K nonpersistent messages, client channels

Figure 28 and **Table 17** show that the throughput of nonpersistent messages has degraded by about 3-4% when comparing Version 6.0 to Version 7.0.

Test name: **clnp_200K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**19** (20)	**325** (322)	**0.0699** (0.0736)	**25%** (25%)
WebSphere MQ V7.0	(19) **20**	(306) **313**	(0.0742) **0.0768**	(29%) **30%**

Table 17 – 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 7.

3.3.2.2 Persistent Messages

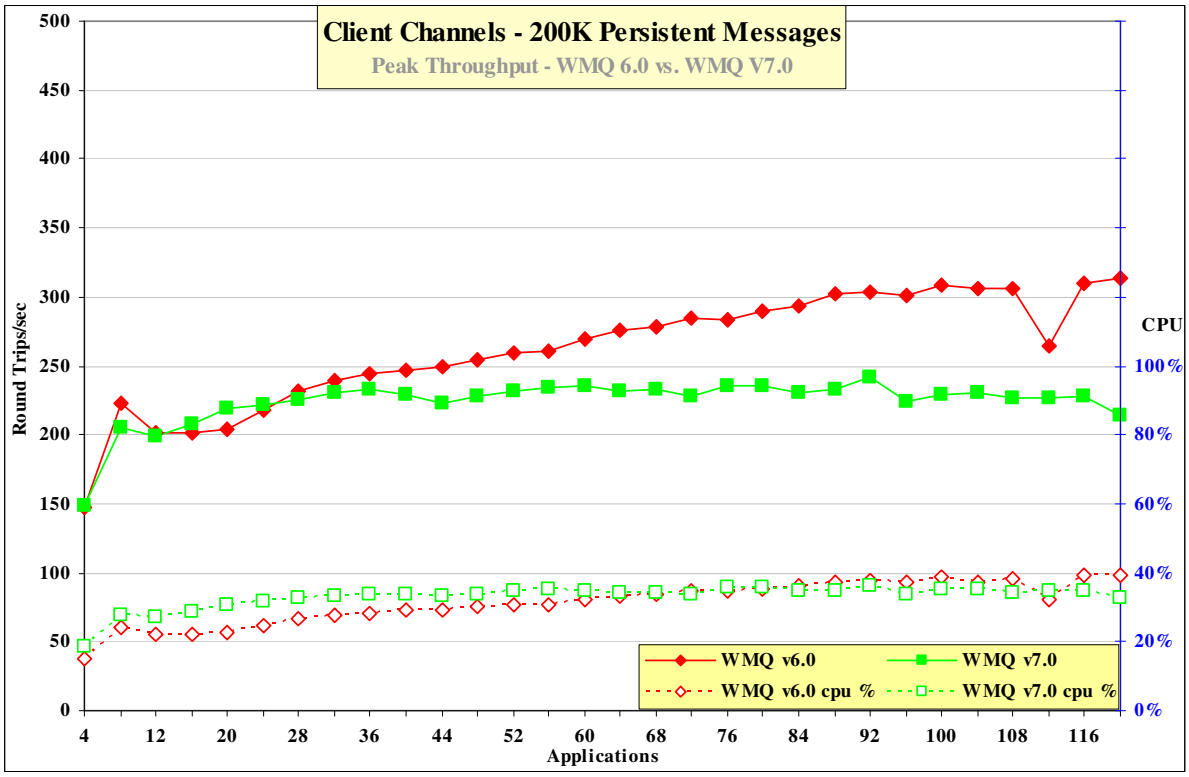


Figure 29 – 200K persistent messages, client channels

Figure 29 and **Table 18** show that that the throughput of nonpersistent messages has degraded by about 23% when comparing Version 6.0 to Version 7.0.

Test name: **clpm_200K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(92) **120**	(304) **313**	(0.3406) **0.4321**	(38%) **40%**
WebSphere MQ V7.0	**92** (120)	**241** (214)	**0.4431** (0.6025)	**36%** (33%)

Table 18 – 200K persistent messages, client channels

3.3.3 Distributed Queuing

Figure 30 and **Figure 31** 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

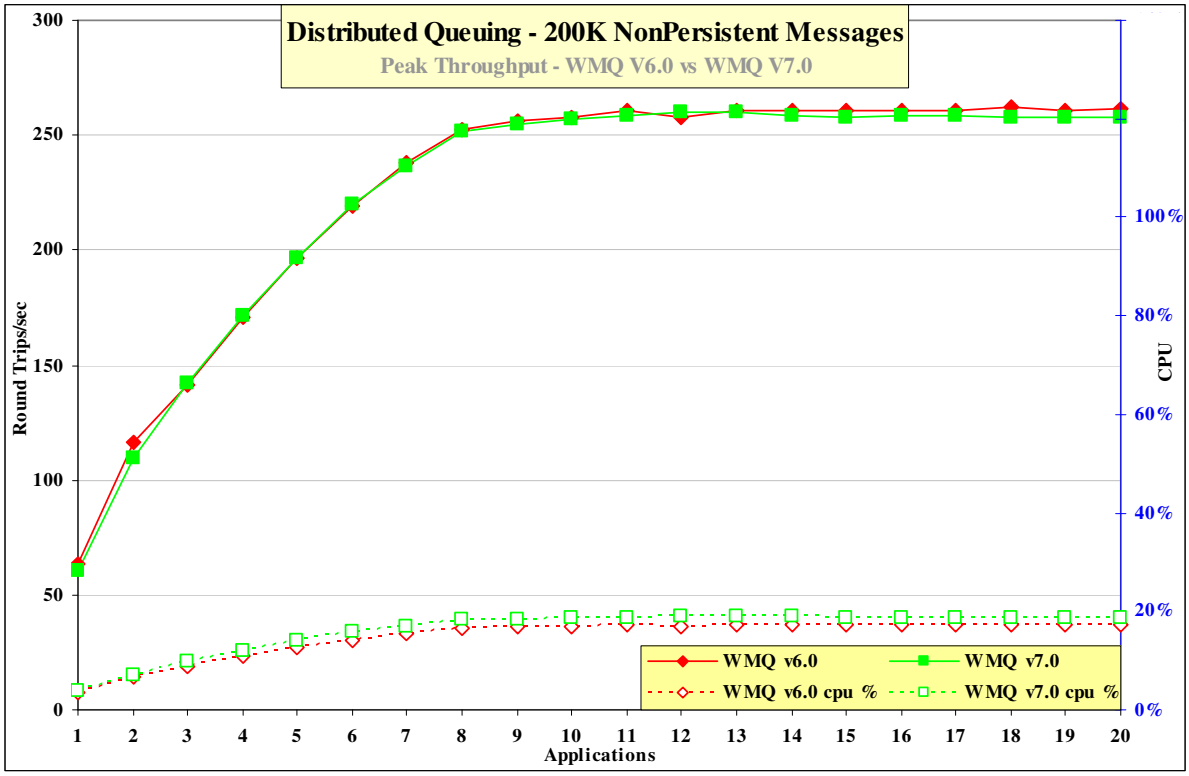


Figure 30 – 200K nonpersistent messages, distributed queuing

Figure 30 and **Table 19** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqnp_200K**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(12 **18**	(258) **262**	(0.0550) **0.0818**	(17%) **17%**
WebSphere MQ V7.0	**12** (18)	**260** (258)	**0.0542** (0.0827)	**19%** (19%)

Table 19 – 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 7.

3.3.3.2 Persistent Messages

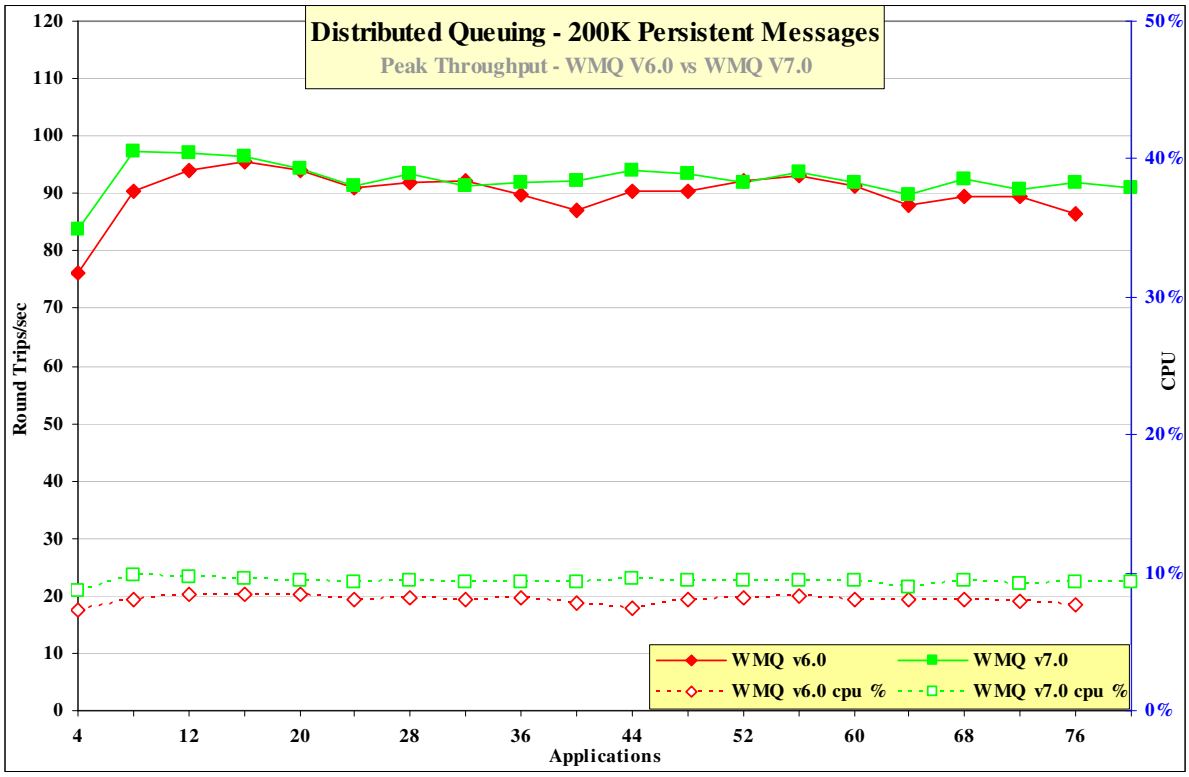


Figure 31 – 200K persistent messages, distributed queuing

Figure 31 and **Table 20** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqpm_200K**	Apps	Round Trips/sec	Response Time	CPU
WebSphere MQ V6.0	(8) **16**	(90) **95**	(0.1012) **0.1887**	(8%) **9%**
WebSphere MQ V7.0	**8** (16)	**97** (96)	**0.0889** (0.1898)	**10%** (10%)

Table 20 – 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 7.

3.4 2MB Messages

3.4.1 Local Queue Manager

Figure 32 and **Figure 33** 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

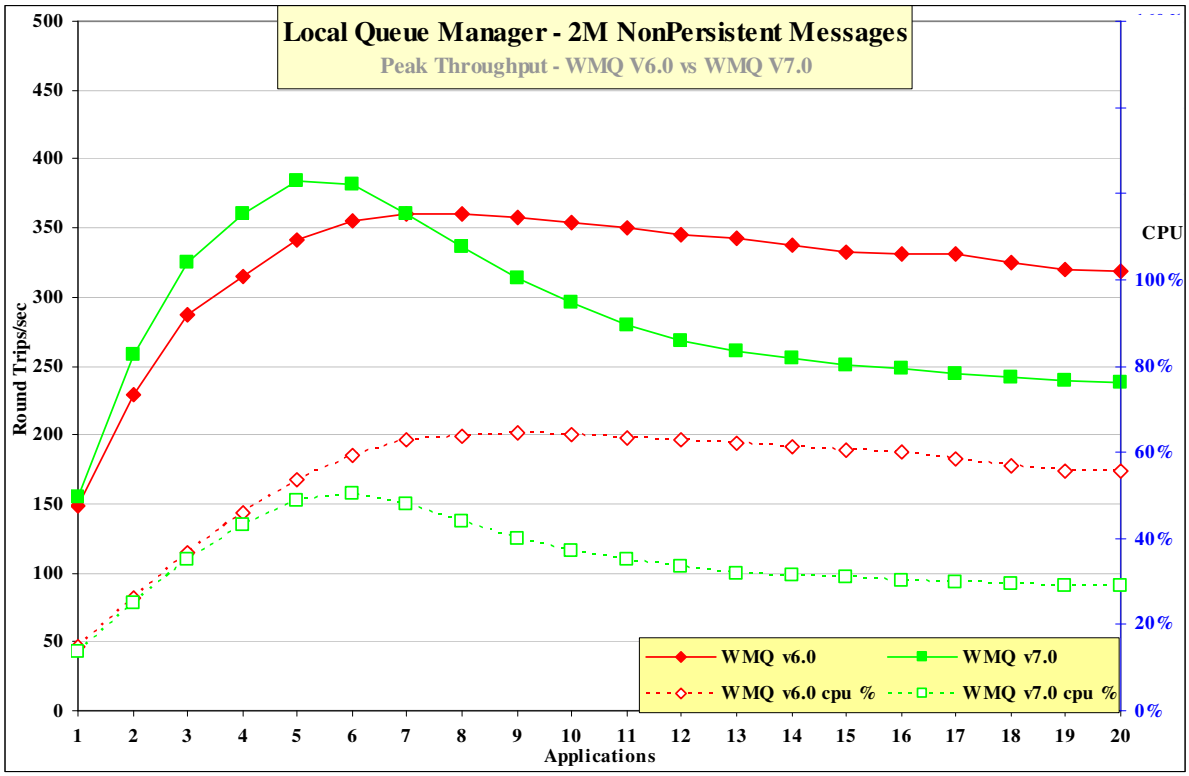


Figure 32 – 2M nonpersistent messages, local queue manager

Figure 32 and **Table 21** show that the maximum throughput level of nonpersistent messages is 6-7% higher when comparing Version 7.0 to Version 6.0.

Test name: **local_np_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(5) **8**	(342) **361**	(0.0164) **0.0254**	(54%) **64%**
WebSphere MQ V7.0	**5** (8)	**384** (336)	**0.0147** (0.0273)	**49%** (44%)

Table 21 – 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 7

3.4.1.2 Persistent Messages

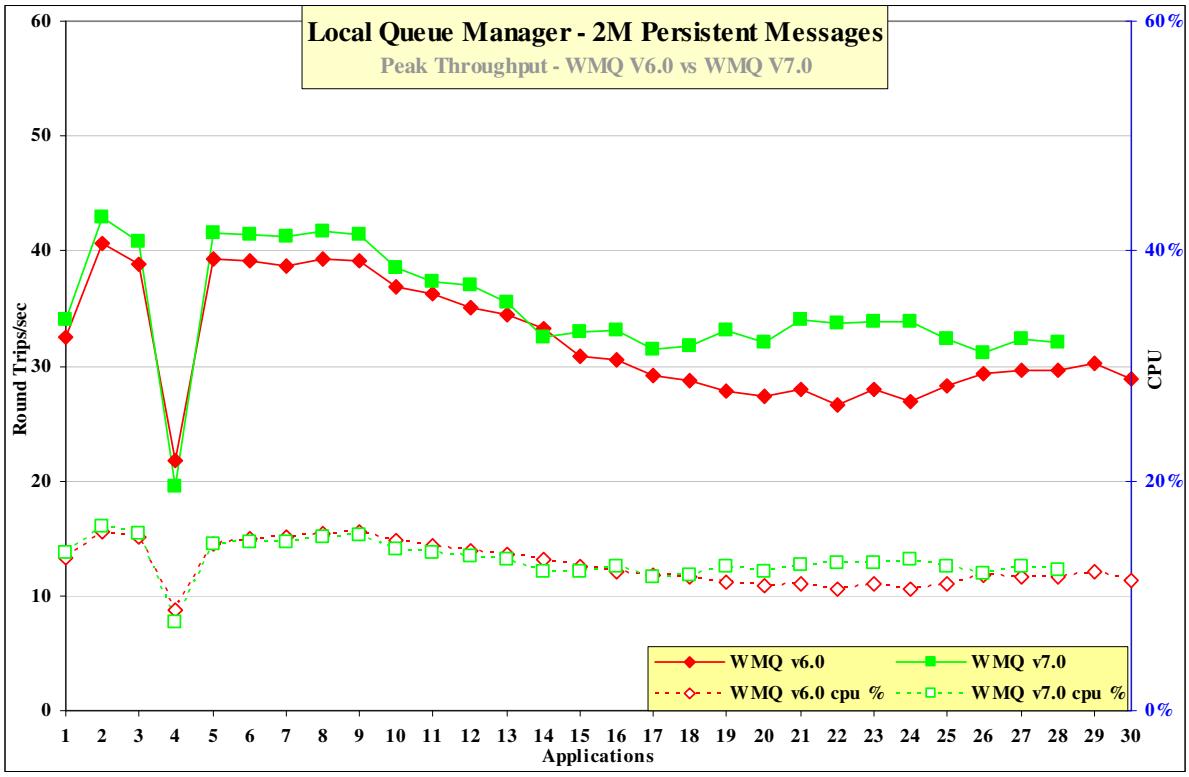


Figure 33 – 2M persistent messages, local queue manager

Figure 33 and **Table 22** show that the throughput of persistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **local_pm_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**2**	**41**	**0.0526**	**16%**
WebSphere MQ V7.0	**2**	**43**	**0.0507**	**16%**

Table 22 – 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 V7.

3.4.2 Client Channel

Figure 34 and **Figure 35** 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

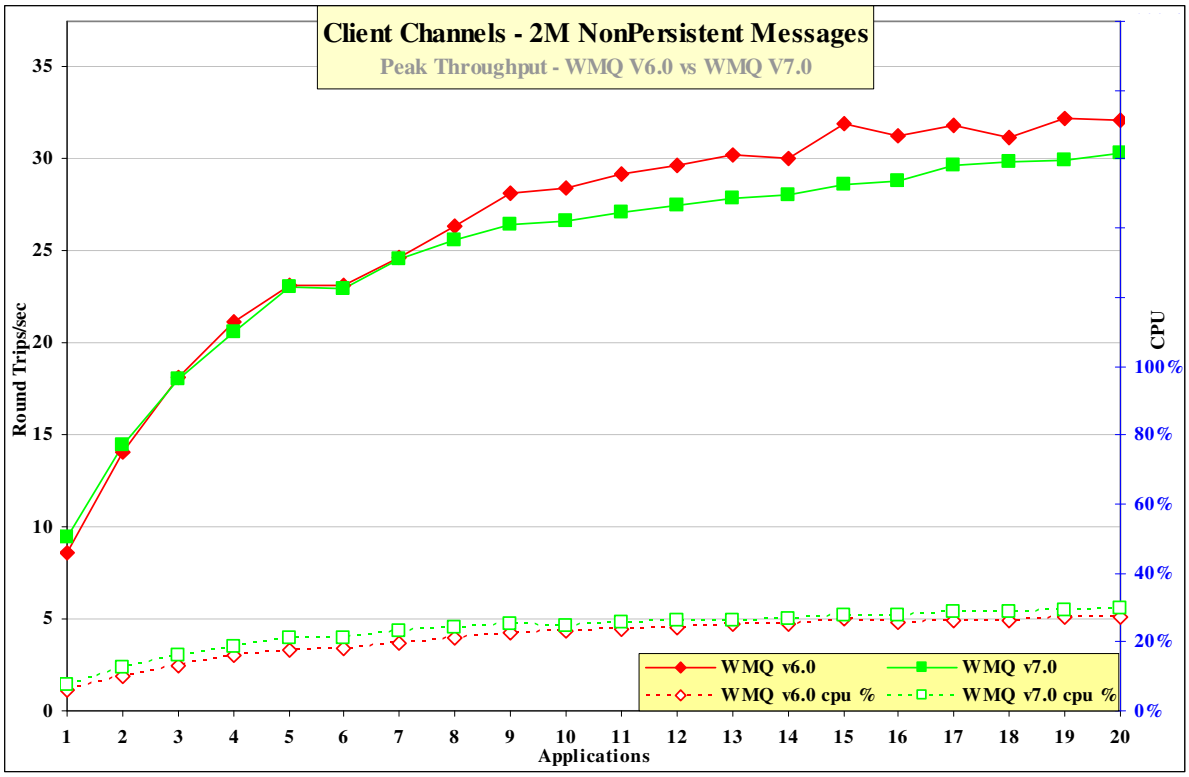


Figure 34 – 2M nonpersistent messages, client channels

Figure 34 and **Table 23** show that the throughput of nonpersistent messages has degraded about 6-7% when comparing Version 6.0 to Version 7.0.

Test name: **clnp_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**19** (20)	**32** (32)	**0.7034** (0.7326)	**27%** (27%)
WebSphere MQ V7.0	(19) **20**	(30) **30**	(0.7589) **0.7900**	(29%) **30%**

Table 23 – 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 7.

3.4.2.2 Persistent Messages

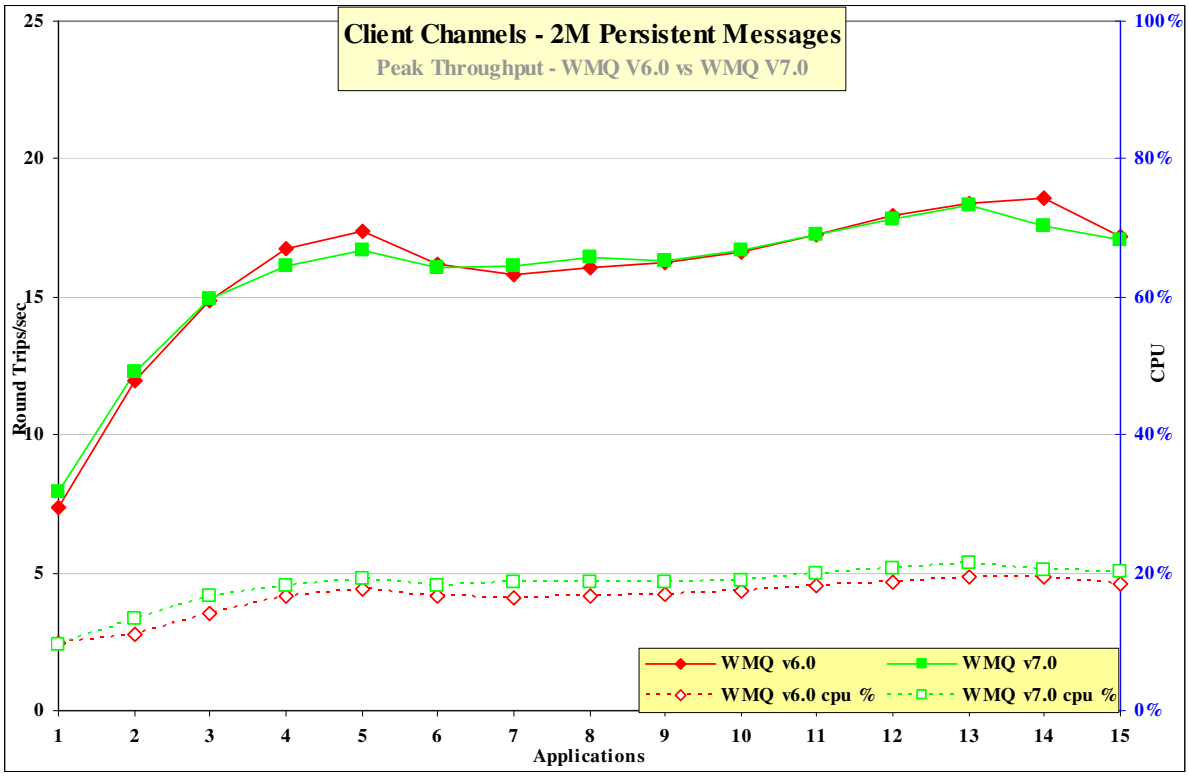


Figure 35 – 2M persistent messages, client channels

Figure 35 and **Table 24** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **clpm_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	(13) **14**	(18) **19**	(0.8347) **0.8928**	(20%) **20%**
WebSphere MQ V7.0	**13** (14)	**18** (18)	**0.8366** (0.9315)	**21%** (20%)

Table 24 – 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 7.

3.4.3 Distributed Queuing

Figure 36 and **Figure 37** 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

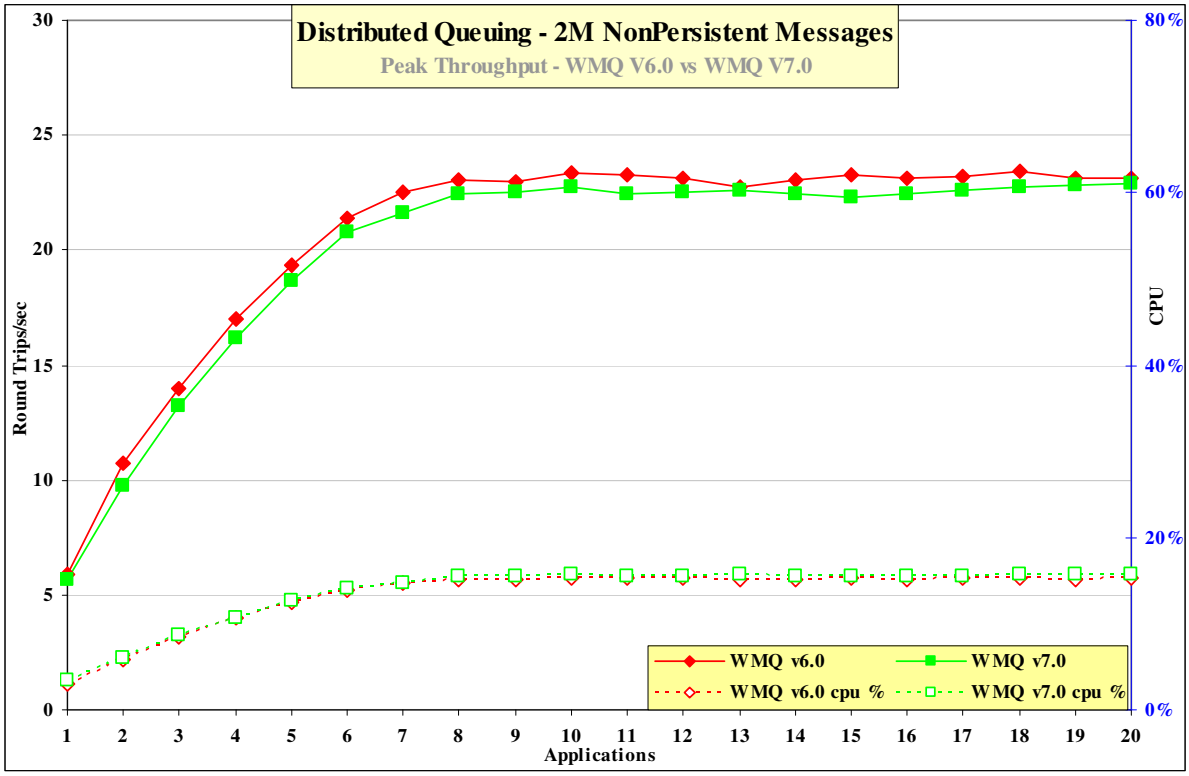


Figure 36 – 2M nonpersistent messages, distributed queuing

Figure 36 and **Table 25** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqnp_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**18** (19)	**23** (23)	**0.9165** (0.9802)	**15%** (15%)
WebSphere MQ V7.0	(18) **19**	(23) **23**	(0.9414) **0.9897**	(16%) **16%**

Table 25 – 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 7.

3.4.3.2 Persistent Messages

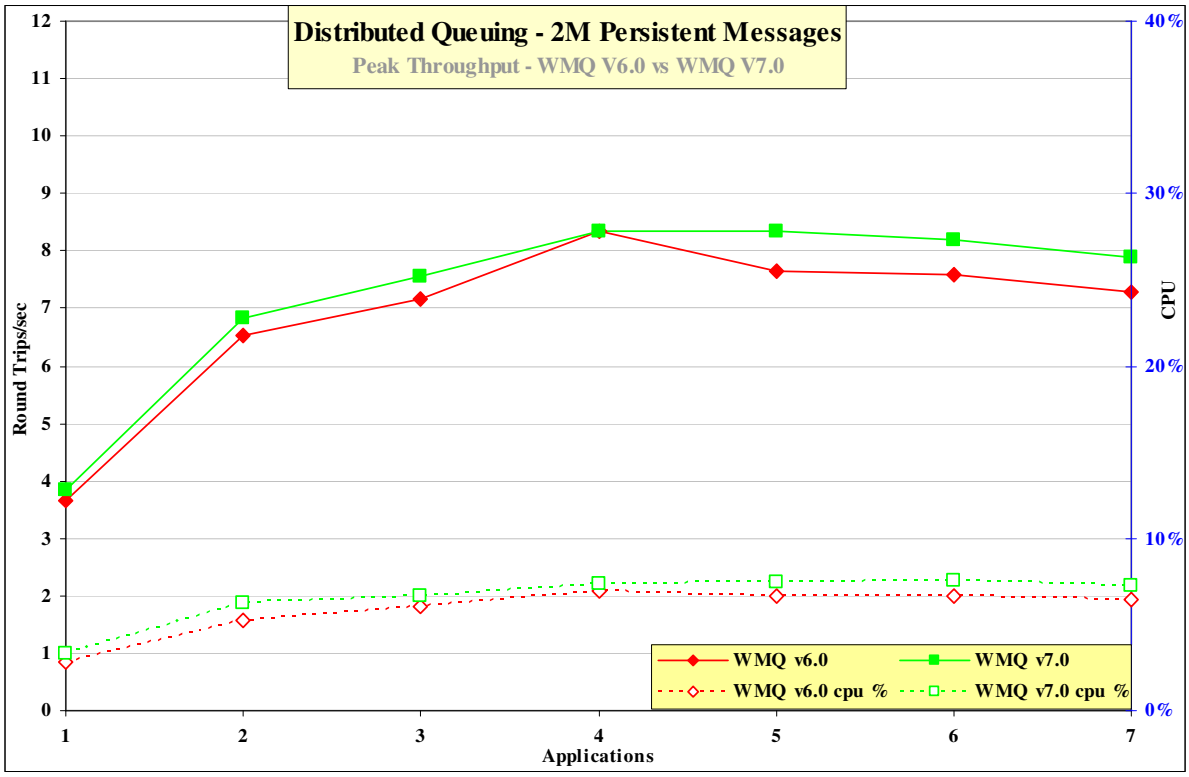


Figure 37 – 2M persistent messages, distributed queuing

Figure 37 and **Table 26** show that the throughput of nonpersistent messages is similar when comparing Version 6.0 to Version 7.0.

Test name: **dqpm_2M**	**Apps**	**Round Trips/sec**	**Response Time**	**CPU**
WebSphere MQ V6.0	**4**	**8**	**0.5609**	**7%**
WebSphere MQ V7.0	**4**	**8**	**0.5639**	**7%**

Table 26 – 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 7.

4 Application Bindings

This report analyzes the rate that messages can be exchanged between a Requester (Driver) application and a Responder (Server) application. This chapter looks at the effect of various combinations of application bindings for Requester and Responder programs.

	Requester	Responder
Normal	Trusted	Non Trusted
Isolated	Isolated	Isolated
Trusted	Trusted	Trusted
Non Trusted	Shared	Shared

4.1 Local Queue Manager

Figure 38 and **Figure 39** 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

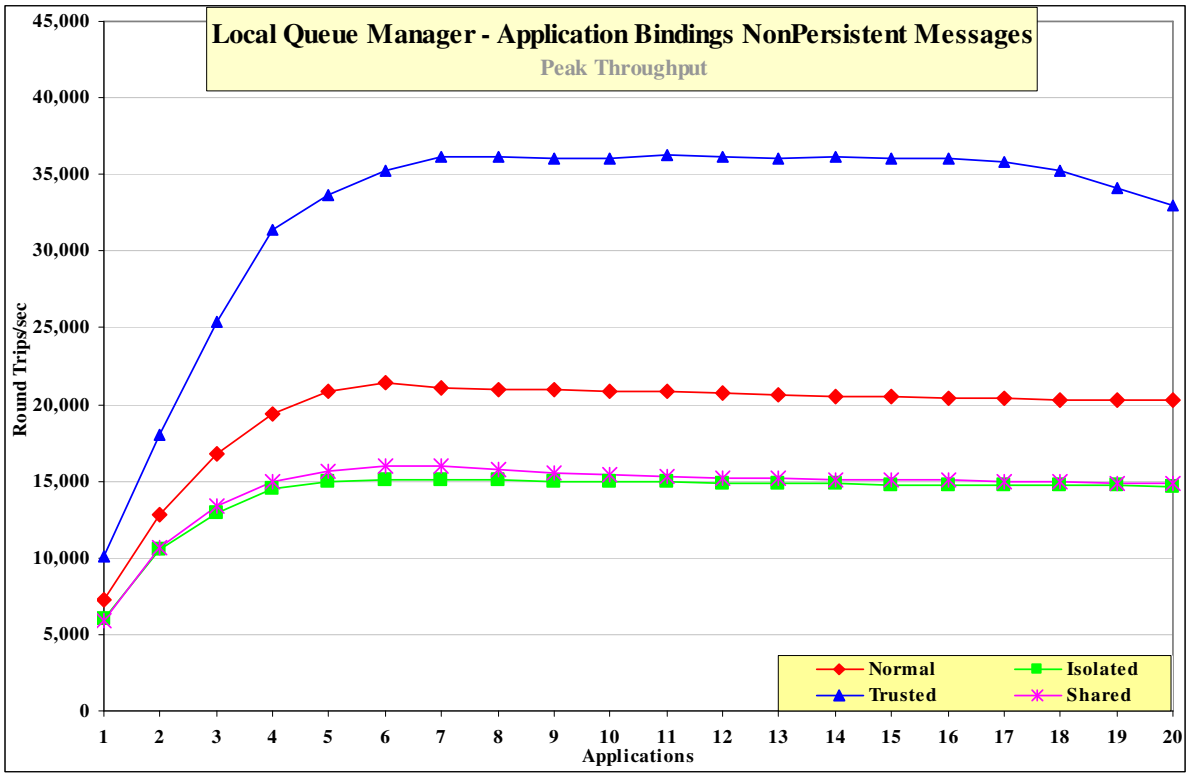


Figure 38 – Application binding, nonpersistent messages, local queue manager

Figure 38 and **Table 27** show the peak throughput of nonpersistent messages when comparing normal, isolated, trusted and shared bindings. Applications using trusted bindings achieve higher throughput levels due to having a much smaller CPU requirement per round trip than the other bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	6	21397	0.0004	99%
Isolated	8	15083	0.0007	99%
Trusted	11	36316	0.0004	99%
Shared	6	16030	0.0005	99%

Table 27 – Application binding, nonpersistent messages, local queue manager

4.1.2 Persistent Messages

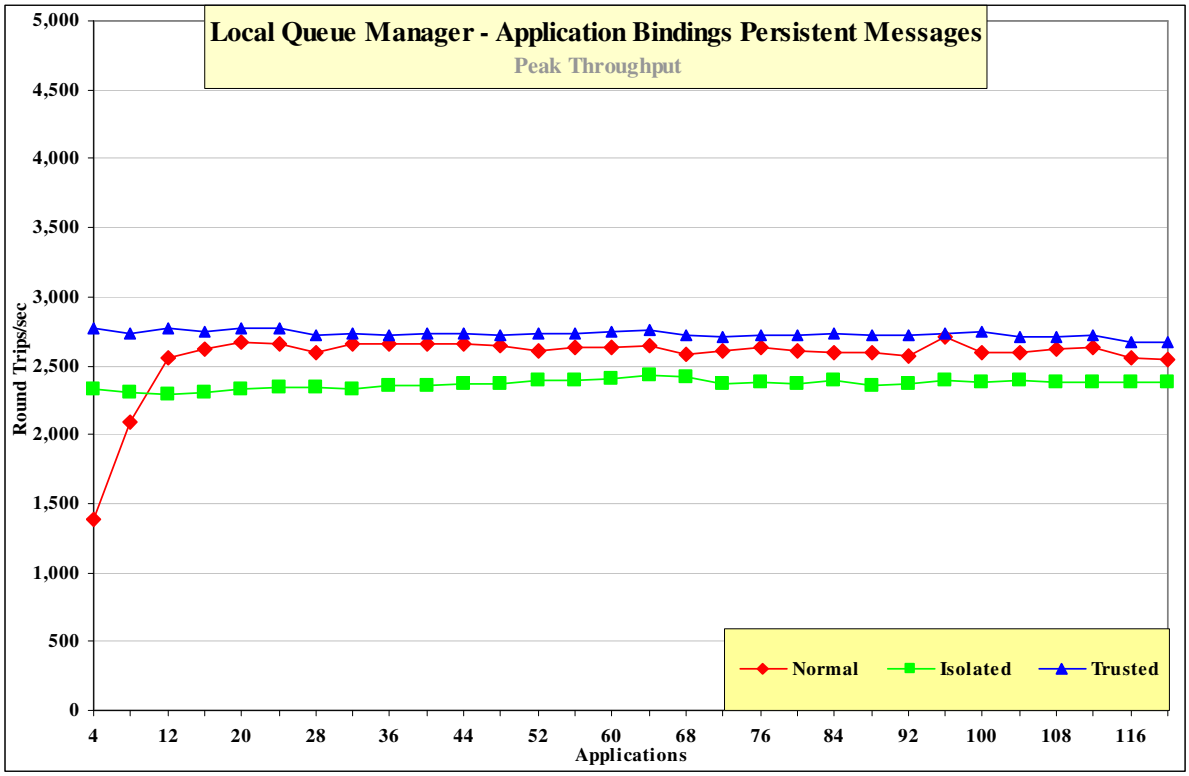


Figure 39 – Application binding, persistent messages, local queue manager

Figure 39 and **Table 28** show the peak throughputs of persistent messages when comparing normal, isolated and trusted bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	**96**	**2708**	**0.0743**	**48%**
Isolated	**64**	**2425**	**0.0441**	**65%**
Trusted	**12**	**2777**	**0.0205**	**46%**

Table 28 – 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 difference in throughput due to using various application bindings is obscured by the I/O necessary for persistent messages. However, the smaller CPU requirement and subsequent higher throughput for trusted bindings can still be observed.

4.2 Client Channels

Figure 40 and **Figure 41** 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

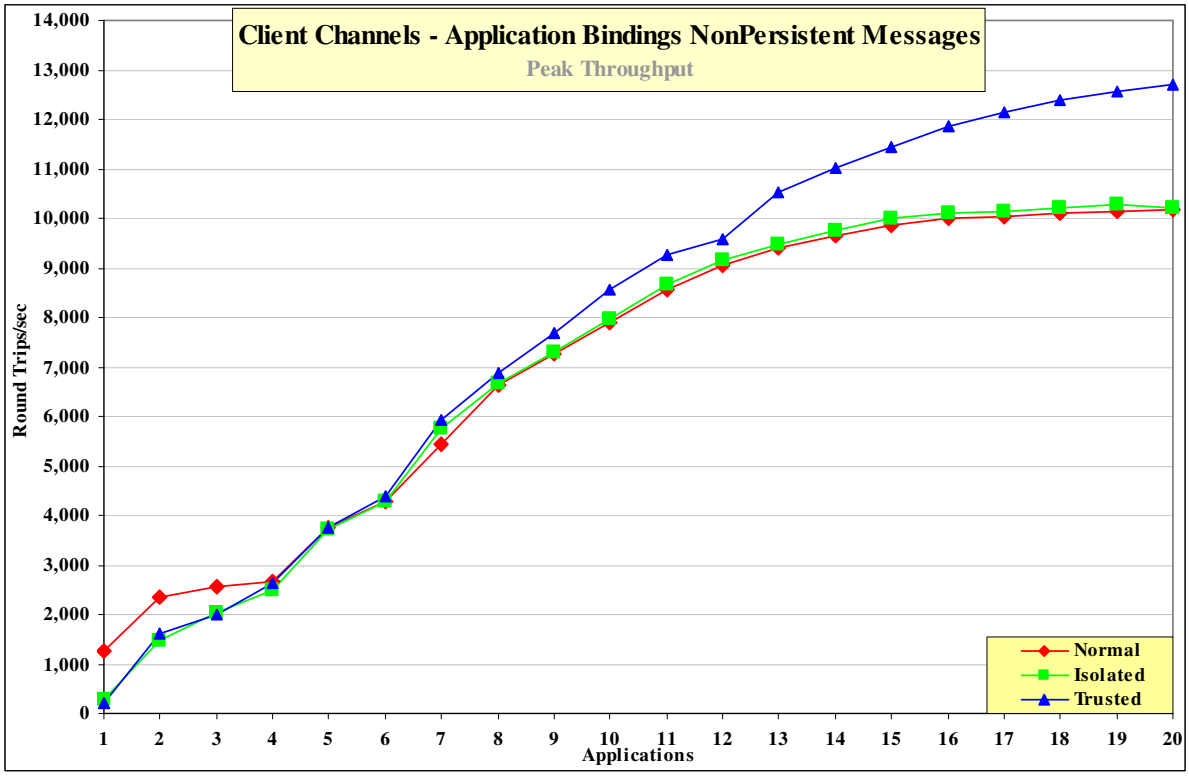


Figure 40 – Application binding, nonpersistent messages, client channels

Figure 40 and **Table 29** show the peak throughput of nonpersistent messages when comparing normal, isolated and trusted bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	**20**	**10173**	**0.0023**	**99%**
Isolated	**19**	**10272**	**0.0022**	**100%**
Trusted	**20**	**12692**	**0.0019**	**97%**

Table 29 – 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.

4.2.2 Persistent Messages

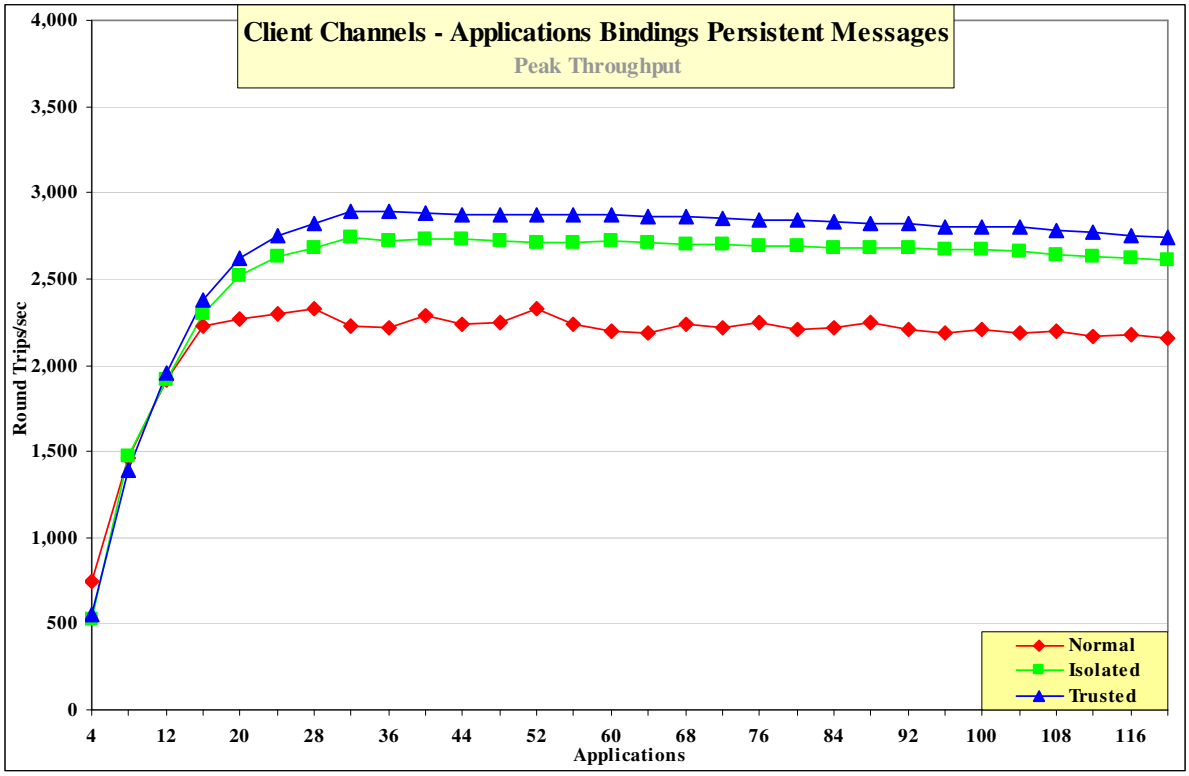


Figure 41 – Application binding, persistent messages, client channels

Figure 41 and **Table 30** show the peak throughput of persistent messages when comparing normal, isolated and trusted bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	**28**	**2330**	**0.0139**	**63%**
Isolated	**32**	**2736**	**0.0137**	**85%**
Trusted	**32**	**2891**	**0.0132**	**78%**

Table 30 – 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 various application bindings for client persistent messages are limited by the same factors as the local persistent messages, thus showing similar behavior in throughput .

4.3 Distributed Queuing

Figure 41 and **Figure 42** 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

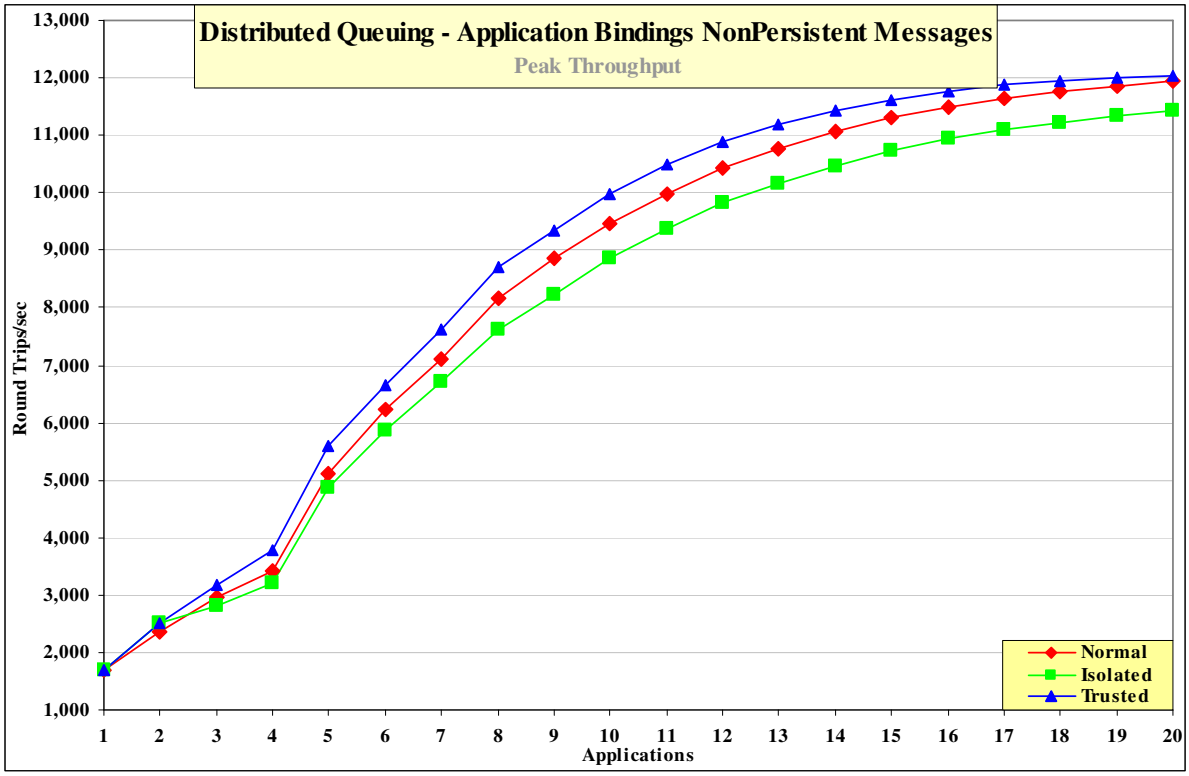


Figure 42 – Application binding, nonpersistent messages, distributed queuing

Figure 42 and **Table 31** show the peak throughput of nonpersistent messages when comparing normal, isolated and trusted bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	**20**	**11942**	**0.0020**	**71%**
Isolated	**20**	**11414**	**0.0022**	**91%**
Trusted	**20**	**12047**	**0.0020**	**46%**

Table 31 – 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.

4.3.2 Persistent Messages

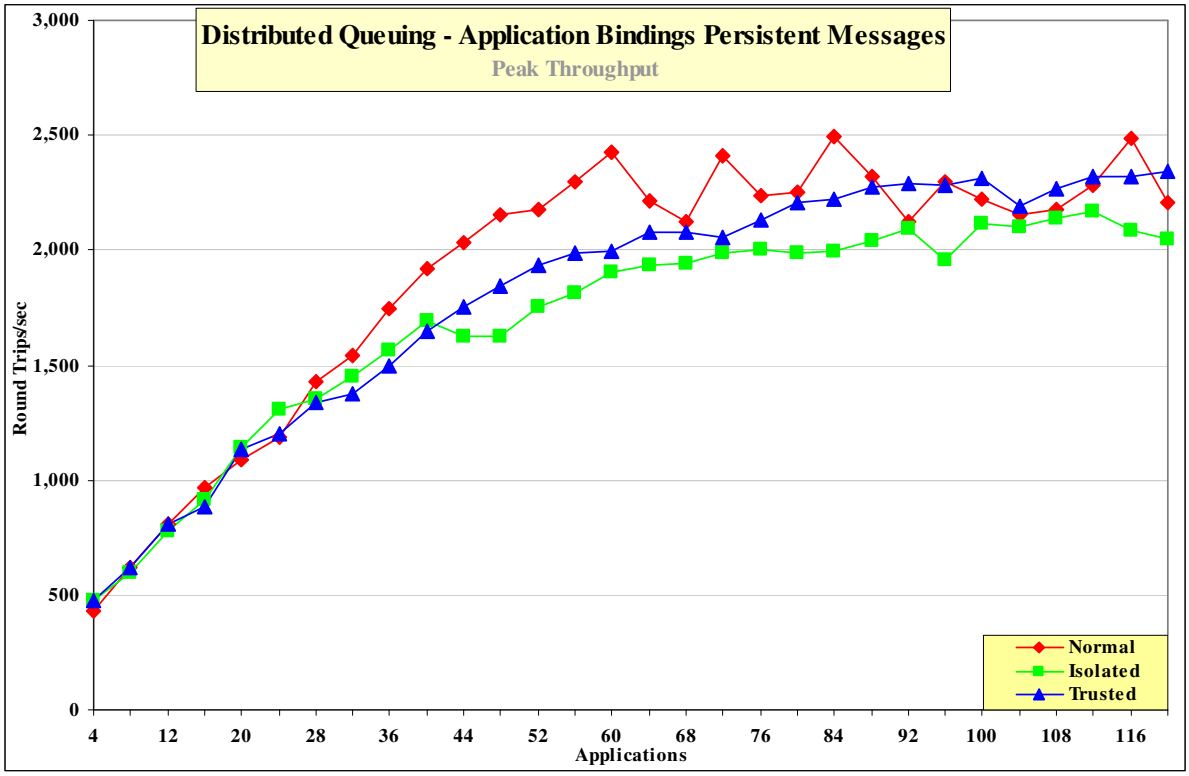


Figure 43 – Application binding, persistent messages, distributed queuing

Figure 43 and **Table 32** show the peak throughput of persistent messages when comparing normal, isolated and trusted bindings.

Bindings	Apps	Round Trips/sec	Response Time	CPU
Normal	**84**	**2496**	**0.0416**	**36%**
Isolated	**112**	**2166**	**0.0618**	**39%**
Trusted	**120**	**2345**	**0.0640**	**28%**

Table 32 – 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 various application bindings for client persistent messages are limited by the same factors as the local persistent messages, thus showing similar behavior in throughput .

5 Short & Long Sessions

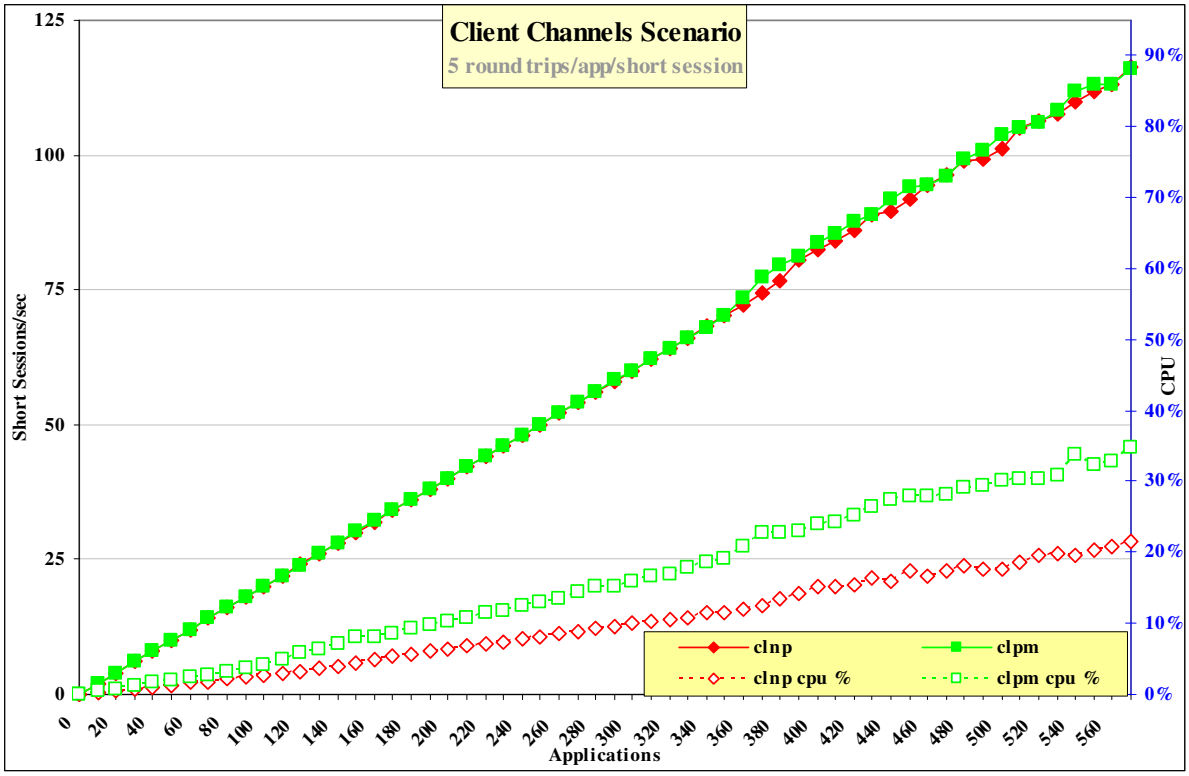
The previous chapters in this report only reported on steady state messaging that does not include any session setup and termination function. This chapter specifically bracket groups of five MQPUT/MQGET pairs with MQCON/MQDISC and MQOPEN/MQCLOSE calls so a comparison of this overhead can be seen.
A 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** (5x)
- **gets the reply message from the common reply queue**
- **wait one second**
- **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 44** for an illustration).

Figure 44 – Short sessions, client channels



Test name	Apps	Round Trips/sec	Short Sessions per second	Response Time (s)	CPU%
clnp_r3600	4500	4499		0.005	64%
clnp_ss (SC0)	1550	1853	371	0.338	73%
clnp_ss (SC1)	570	582	117	0.067	22%
clpm_r3600	2100	2097		0.473	70%
clpm_ss (SC0)	910	983	197	0.229	54%
clpm_ss (SC1)	570	580	116	0.130	35%

Table 33 – 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*

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.* The maximum number of connected applications is likely to be determined by other criteria such as recovery time or manageability. 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. The enhanced client architecture of V7 multiplexes many connects from the same process onto the same server socket. These measurements ignored this facility by either running in V6 compatability mode (SC0 sharecnv=0 on svrcon channel) or setting each client to its own socket (SC1 sharecnv=1 on svrcon channel).

Queue manager configuration for client channels capacity tests:

MaxChannels=50000 (100,000 for clnp_cmax)

Test name	Apps	Rate/app/hr	Round Trips/sec	Response time (s)	CPU
clnp	20	n/a*	10173	0.002	99%
clnp_r3600	4500	3600	4499	0.005	64%
clnp_c6000 (SC0)	6000	4440	7394	0.004	78%
clnp_c6000 (SC1)	6000	2460	4097	0.029	62%
clnp_c6000_nocorrelid (SC0)	6000	3030	5047	0.008	81%
clnp_c6000_nocorrelid (SC1)	6000	2790	4649	0.088	89%
clnp_cmax (SC0)	28250	60	470	0.004	6%
clnp_cmax (SC1)	23750	60	396	0.244	21%
clnp_cmax_nocorrelid (SC0)	28200	60	470	0.004	22%
clnp_cmax_nocorrelid (SC1)	28200	60	470	0.006	24%
clpm_c6000 (SC1)	6000	750	1250	0.824	45%

Table 34 – Capacity measurements, client channels

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

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=4096, LogBufferPages=256
```

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	20	n/a*	11942	0.002	71%
dqnp_r3600	5000	3600	4998	0.002	28%
dqnp_qmax	3300	60	55	0.003	1%
dqnp_q1000	1000	17680	4799	0.104	38%
dqpm_q1000	1000	1440	384	0.397	12%

Table 35 – Capacity measurements, server channels

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

The dqnp and dqnp_r3600 tests both used a total of 4 pairs of Sender/Receiver pairs of channels between queue managers while the dqnp_q1000, dqpm_2000, and dqnp_qmax tests used a pair of channels per application. The dqnp_q1000 test shows the reduced throughput experienced when 1000 queue managers are connected into a central hub.

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 V7.0; some of these can be applied to Version 6. 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

Nonpersistent messages are held in main memory, spilt to the file system as the queues become deep, and lazily written to the Queue file. Persistent messages are synchronously written to the log by an MQCmit and also periodically flushed to the Queue file.

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. Multiple disks can be redirected to a Storage Area Network (SAN) but multiple high volume Queue managers can require different Logical Volumes to avoid congestion.

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 minimized by using cached disks or SAN.

7.1.1.1 Nonpersistent and Persistent Queue Buffer

The default nonpersistent queue buffer size is 64K per queue, and the default for persistent is 128K per queue for 32 bit queue managers and 128K /256K for 64 bit queue managers (IBM i, Solaris, HPUX, Linux_64, z_Linux, and Windows64). They can all 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 current setting. When a queue is opened, resources are allocated according to the definition held on disk from when the queue was created.

7.1.2 Channels: Process or Thread, Standard or Fastpath?

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.

7.1.3 Multiplexed Clients

Previous levels (including Version 6) of WebSphere MQ used a separate TCP socket for each client. Version 7 will multiplex clients from the same process over one TCP socket. Chapter 2 show the difference in performance of these variants. Version 6 behavior can be obtained by using the 'sharecnv' keyword with a setting of zero, as in this example:

```
define channel( csim_channel_TCP ) +
        chltype( svrconn ) +
        trptype( tcp ) +
        sharecnv( 0 )
```

Note that using Version 6 behavior will also inhibit new performance features of V7 like 'ASYNC Put' and 'READ_AHEAD'.

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 as 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 utilized 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.

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.

7.2.3 Persistent Messaging Considerations

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.

Journal Considerations on IBM i

Persistent messaging on IBM i uses native journaling support to ensure that messages are recoverable. To ensure that maximum rates of throughput are achieved when using persistent messages, you should consider the following:

When the queue manager is created on IBM i, a journal receiver is automatically created and located on the system ASP disk arms. To avoid contention with other IO on these arms, it is recommended that the user manually create and attach a new journal receiver, ensuring that it is located on a user ASP with dedicated disk arms. This will help improve response times for the synchronous disk writes to the journal that are needed for each persistent message.

It will also be helpful to ensure that the disk arms and IOPs used in the user ASP have good overall performance characteristics for write activity, including good write cache performance. Slower disk arms and IOPs will result in less favorable response times and less overall capacity in terms of message throughput.

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 64 bit libraries for MQ V6 & V7.

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 2**.

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, and 6 threads respectively for processing nonpersistent messages with local, client, and distributed applications, and 30, 60, and 10 threads respectively to process persistent messages with local, client, and distributed applications. The capacity tests in chapter 5 and 6 use 10 server threads for processing non persistent messages. Each server thread performed the sequence of actions as outlined in the test scenario illustrations in the "**Performance Headlines**" starting on **page 2**.

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

Server system:	IBM iSeries
Model:	550
Processor:	1.9GHz Power 5
Architecture:	4-way SMP
Memory (RAM):	32GB
Disk:	10 70GB disk arms (system ASP, no user ASPs configured)
Network:	1GBit Ethernet Adapter

Driver system:	IBM xSeries 365
Model:	8862-6RX
Processor:	Intel Pentium 4 Xeon 3 GHz, 4MB L3 Cache, non-Hyperthreaded
Architecture:	4-way SMP
	IBM SerialRAID Adapter
Memory (RAM):	4GB
Disk:	2 Internal Ultra320 SCSI (72GB each, 1 O/S, swap)
	2 SCSI disks (72GB each, 1 queue, 1 log)
Network:	1GBit Ethernet Adapter

8.3 Software

IBM i O/S:	IBM i V5R4M0
MQSeries:	Version 7.0, Version 6.0
Compiler:	C for IBM i 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.*
Response Time	The average response time each round trip in seconds, 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.
SC0	SHARECNV=0 specified on the def channel(x) chltype(svrconn) command. Version 6 compatibility mode
SC1	SHARECNV=1 specified on the def channel(x) chltype(svrconn) command. Separate socket per client