

WebSphere MQ for Solaris V6.0 Sun Fire T2000 Performance Evaluations

Version 1.3

May 2006

Andrew Walton
Market Development Engineering
Sun Microsystems

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

Property of IBM

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.

Fourth Edition, May 2006.

This edition applies to *WebSphere MQ V6.0 for Solaris on Sun Fire T2000* (and to all subsequent releases and modifications until otherwise indicated in new editions).

© Copyright International Business Machines Corporation 2006. All rights reserved.

Note to U.S. Government Users

Documentation related to restricted rights.

Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

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 V6.0 for Solaris on Sun Fire T2000*. The information is not intended as the specification of any programming interface that is provided by WebSphere MQ. It is assumed that the reader is familiar with the concepts and operation of WebSphere MQ V6.

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 MQ
- DB2

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

The following term, used in this publication is a trademark of the SUN Microsystems Corporation:

Solaris

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

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

EXPORT REGULATIONS

WebSphere MQ for Solaris V6.0 – T2000 Performance Evaluations

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 environments 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 V** to familiarise themselves with where specific information can be found for later reference.

The Contents of this SupportPac

This SupportPac includes:

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

Specific queries about performance problems on your WebSphere MQ system should be directed to your local IBM representative or Support Center.

Introduction

The three scenarios in this report used to generate the performance data are classified into: the local queue manager scenario, the client channel scenario, and the distributed queuing scenario.

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

A Sun Fire T2000 running Solaris 10 was used as the server device under test for all the measurements in this report.

How this document is arranged

Performance headlines

Pages:1 - 17

Section one of the document contain the performance *headlines* for each of the three scenarios, with MQI applications connected to:

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

The headline tests show:

- the *maximum* message throughput achieved with an increasing number of MQI applications,
- the *maximum* number of MQI-clients connected to a queue manager, and
- 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.

These tests were performed with 16, 32, 48 and 64 sever threads. All four sets of results are presented in order to show when increasing or decreasing the number of server threads will improve message throughput.

Persistent message throughput was measured using the T2000's internal SAS (Serial Attach SCSI) disks and with an external 3510 fibre channel RAID array which has a 1G cache.

When using the internal disks, one drive was dedicted to the queue manager and a second drive was dedicated to the logs. The 3510 was configured to expose two RAID 0 LUN's, each containg 6 disks, one LUN was used for the queue manager and the second was used for the logs.

Solaris Zones are a software partitioning technology used to virtualize operating system services and provide isolated and secure environments for running applications. Zones are

typically used for server consolidation. The isolation provided by zones prevents processes that are running in one zone from monitoring or effecting processes running in another zone. The only way for processes in different zones to communicate is via a network connection. Since the processes are on the same host, this network connection is optimised and the connection is made in memory and not over a physical network.

Large messages

Pages: 18 - 56

Section two of the document contains performance measurements for *large messages of 20K and 200K* messages using the same scenarios as for the performance *headlines*. In addition throughput of messages size from 1K to 256K in powers of 2 with 32 server threads are also shown.

Short sessions

Pages: 57 - 58

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

Capacity measurements

Pages: 59 -60

Section four of this document shows:

- the total number of MQI-client channels that were connected into a single queue manager, with a server application processing 1 nonpersistent round trip per MQI-client per *minute*.
- the total number of server channel pairs that were connected between two queue managers on separate server machines, with a server application processing 1 nonpersistent round trip per server channel pair per *minute*.

Tuning recommendations

Pages: 61 - 64

In previous SupportPacs tuning recommendations have been in a separate section. In this document queue manger parameters are mentioned with the measurements they are appropriate to, with detailed discussion in pages 61 - 64

Measurement environment

Pages: 65 - 67

A summary of the way in which the workload is used in each test scenario is given in the performance headlines section. For a more detailed description of the workload, hardware and software specifications, refer to the pages 65 - 67

Glossary

Page: 68

A short glossary of the terms used in the tables throughout this document can be found in the 'Glossary'—Page 68.

Contents

1	Performance headlines.....	1
1.1	Local queue manager test scenario.....	2
1.1.1	Non persistent messages – local queue manager.....	3
1.1.2	Persistent messages – local queue manager – internal disks.....	4
1.1.3	Persistent messages – local queue manager – external disks.....	5
1.2	Client channels test scenario between hosts.....	6
1.2.1	Non persistent messages – client channels between hosts.....	7
1.2.2	Persistent messages – client channels between hosts– internal disks.....	8
1.2.3	Persistent messages – client channels between hosts – external disks.....	9
1.3	Client channels test scenario between zones.....	10
1.3.1	Non persistent messages – client channels between zones.....	11
1.3.2	Persistent messages – client channels between zones – internal disks.....	12
1.3.3	Persistent messages – client channels between zones – external disks.....	13
1.4	Distributed channels test scenario.....	14
1.4.1	Non persistent messages – server channels.....	15
1.4.2	Persistent messages – server channels – internal disks.....	16
1.4.3	Persistent messages – server channels – external disks.....	17
2	Large Messages.....	18
2.1	Local queue manager test scenario – large messages.....	19
2.1.1	Non persistent large messages – local queue manager.....	20
2.1.2	Persistent large messages – local queue manager – internal disks.....	23
2.1.3	Persistent large messages – local queue manager – external disks.....	26
2.2	Client channels test scenario between hosts.....	29
2.2.1	Non persistent large messages – client channels between hosts.....	30
2.2.2	Persistent large messages – client channels between hosts– internal disks.....	33
2.2.3	Persistent large messages – client channels between hosts – external disks...	36
2.3	Client channels test scenario between zones.....	39
2.3.1	Non persistent large messages – client channels between zones.....	40
2.3.2	Persistent large messages – client channels between zones – internal disks...	43
2.3.3	Persistent large messages – client channels between zones – external disks.	44
2.4	Distributed channels test scenario.....	47
2.4.1	Non persistent large messages – server channels.....	48
2.4.2	Persistent large messages – server channels – internal disks.....	51
2.4.3	Persistent large messages – server channels – external disks.....	54
3	Short sessions.....	57
4	Performance and capacity limits.....	59
4.1	Client channels scenario – capacity measurements.....	59
4.2	Distributed queuing scenario – capacity measurements.....	59
5	Tuning recommendations.....	61
5.1	Tuning the queue manager.....	61
5.1.1	Queue disk, Log disk, and message persistence.....	61
	Nonpersistent queue buffer.....	61
5.1.2	Log buffer size, Log file size, and number of log extents.....	62
5.1.3	Channels: process or thread, standard or fastpath?.....	62
5.2	Application design and configuration.....	63
5.2.1	Standard or fastpath?.....	63
5.2.2	Parallelism, batching, and triggering.....	63

WebSphere MQ for Solaris V6.0 – T2000 Performance Evaluations

5.3 Tuning the Solaris 10 Operating System.....64

6 Measurement environment.....65

6.1 Workload description.....65

6.1.1 MQI response time tool.....65

6.1.2 Test scenarios workload.....65

6.2 Hardware.....67

6.3 Software.....67

7 Glossary.....68

Figures

Figure 1: Connections into a local queue manager.....	2
Figure 2: Non persistent 2K messages, local queue manger.....	3
Figure 3: Persistent 2K messages, local queue manager, internal disk.....	4
Figure 4: Persistent 2K messages, local queue manager, external disk.....	5
Figure 5: MQI-client channels into a remote queue manager.....	6
Figure 6: Non persistent 2K messages, client channels between hosts.....	7
Figure 7: Persistent 2K messages, client channels between hosts, internal disks.....	8
Figure 8: Persistent 2K messages, client channels between hosts, external disks.....	9
Figure 9: MQI client channels into a remote queue manager using zones.....	10
Figure 10: Non persistent 2K messages, client channels between zones.....	11
Figure 11: Persistent 2K messages, client channels between zones, external disks.....	13
Figure 12: Server channels between two queue managers.....	14
Figure 13: Non persistent 2K messages, server channels.....	15
Figure 14: Persistent 2K messages, server channels, internal disks.....	16
Figure 15: Persistent 2K messages, server channels, external disks.....	17
Figure 16: Non persistent 20K messages, local queue manager.....	21
Figure 17: Non persistent 200K messages, local queue manager.....	21
Figure 18: Comparison of non persistent 2K, 20K & 200K messages, local queue manager	22
Figure 19: Comparison of non persistent messages from 1K to 256K, local queue manager	22
Figure 20: Persistent 20K messages, local queue manager, internal disks.....	24
Figure 21: Persistent 200K messages, local queue manager, internal disks.....	24
Figure 22: Comparison of 2, 20K and 200K persistent messages, local queue manager, internal disks.....	25
Figure 23: Comparisons of persistent messages from 1K to 256K, local queue manager, internal disks.....	25
Figure 24: Persistent 20K messages, local queue manager, external disks.....	27
Figure 25: Persistent 200K messages, local queue manager, external disks.....	27
Figure 26: Comparison of 2K, 20K & 200K persistent messages, local queue manager, external disks.....	28
Figure 27: Comparison of persistent messages from 1K to 256K, local queue manager, external disks.....	28
Figure 28: Non persistent 20K messages, client channels between hosts.....	31
Figure 29: Non persistent 200K messages, client channels between hosts.....	31
Figure 30: Comparison of 2K, 20K & 200K non persistent messages, client channels between hosts.....	32
Figure 31: Comparison of non persistent messages from 1K - 256K, client channels between hosts.....	32
Figure 32: Persistent 20K messages, client channels between hosts, internal disks.....	34
Figure 33: Persistent 200K messages, client channels between hosts, internal disks.....	34
Figure 34: Comparison of 2K, 20K & 200K persistent messages, client channels between hosts, internal disks.....	35
Figure 35: Comparison of persistent messages from 1K to 256K, client channels between hosts, internal disks.....	35
Figure 36: Persistent 20K messages, client channels between hosts, external disks.....	37
Figure 37: Persistent 200K messages, client channels between hosts, external disks.....	37

Figure 38: Comparison of 2K, 20K & 200K persistent messages, client channels between hosts, external disks.....38

Figure 39: Comparison of persistent messages from 1K to 256K, client channels between hosts, external disks.....38

Figure 40: Non persistent 20K messages, client channels between zones.....41

Figure 41: Non persistent 200K messages, client channels between zones.....41

Figure 42: Comparison of 2K, 20K & 200K non persistent messages, client channels between zones.....42

Figure 43: Comparison of non persistent messages from 1K to 256K, client channels between zones.....42

Figure 44: Persistent 20K messages, client channels between zones, external disks.....45

Figure 45: 200K persistent messages, client channels between zones, external disks.....45

Figure 46: Comparison of 2K, 20K & 200K persistent messages, client channels between zones, external disks.....46

Figure 47: Comparison of persistent messages from 1K to 256K, client channels between zones, external disks.....46

Figure 48: Non persistent 20K messages, server channels.....49

Figure 49: Non persistent 200K messages, server channels.....49

Figure 50: Comparison of non persistent 2K, 20K & 200K messages, server channels...50

Figure 51: Comparison of non persistent messages from 1K to 256K, server channels...50

Figure 52: Persistent 20K messages, server channels, internal disks.....52

Figure 53: Persistent 200K messages, server channels, internal disks.....52

Figure 54: Comparison of 2K, 20K & 200K persistent messages, server channels, internal disks.....53

Figure 55: Comparison of persistent messages from 1K to 256K, server channels, internal disks.....53

Figure 56: Persistent 20K messages, server channels, external disks.....55

Figure 57: Persistent 200K messages, server channels, external disks.....55

Figure 58: Comparison of 2K, 20K & 200K persistent messages, server channels, external disks.....56

Figure 59: Comparison of persistent messages from 1K to 256K, server channels, external disks.....56

Figure 60: Short sessions.....58

Figure 61: Hardware configuration.....67

Tables

Table 1: Peak non persistent 2K message performance, local queue manager.....	3
Table 2: Peak persistent 2K message performance, local queue manager, internal disks. .	4
Table 3: Peak persistent 2K messages performance, local queue manager, external disks	5
Table 4: Peak non persistent 2K messages, client channels between hosts.....	7
Table 5: Peak persistent 2K message performance, client channels between hosts, internal disks.....	8
Table 6: Peak persistent 2K message performance, client channels between hosts, external disks.....	9
Table 7: Peak non persistent 2K message performance, client channels between zones .	11
Table 8: Peak persistent 2K message performance, client channels between zones, external disks.....	13
Table 9: Peak non persistent 2K message performance, server channels.....	15
Table 10: Peak persistent 2K message performance, server channels, internal disks.....	16
Table 11: Peak persistent 2K message performance, server channels, external disks.....	17
Table 12: Peak non persistent 2K, 20K & 200K message performance, local queue manager.....	20
Table 13: Peak persistent 2K, 20K & 200K message performance, local queue manager, internal disks.....	23
Table 14: Peak persistent 2K, 20K & 200K message performance, local queue manager, internal disks.....	26
Table 15: Peak non persistent 2K, 20K & 200K message performance, client channels between hosts.....	30
Table 16: Peak persistent 2K, 20K & 200K message performance, client channels between hosts, internal disks.....	33
Table 17: Peak persistent 2K, 20K & 200K message performance, client channels between hosts, external disks.....	36
Table 18: Peak non persistent 2K, 20K & 200K message performance, client channels between zones.....	40
Table 19: Peak persistent 2K, 20K & 200K message performance, client channels between zones, external disks.....	44
Table 20: Peak non persistent 2K, 20K & 200K message performance, server channels .	48
Table 21: Peak persistent 2K, 20K and 200K message performance, server channels, internal disks.....	51
Table 22: Peak persistent 2K, 20K & 200K message performance, server channels, external disks.....	54
Table 23: Peak short session rate.....	57
Table 24: Client channel capacity measurement.....	59
Table 25: Distributed queuing capacity measurements.....	59
Table 26: Distributed queuing capacity measurements.....	60
Table 27: Distrubuted queuing capacity, swap and shared memory requirements.....	60
Table 28: Software.....	67

1 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*.

Non persistent message tests are terminated after 64 driving applications have connected, while persistent message tests are terminated after 256 driving applications have connected. Both types of test are automatically terminated if the response time exceeds 1 second, this can be seen in the very large (200K) persistent message tests which typically terminate before 128 driving applications have connected.

In the rated messaging tests, the rate used is 1 round trip per driving application per *second*. In the client channel test scenarios, each driving application uses a dedicated MQI-client channel, and in the distributed queuing test scenarios, one or more applications submit messages over a fixed number of server channels.

1.1 Local queue manager test scenario

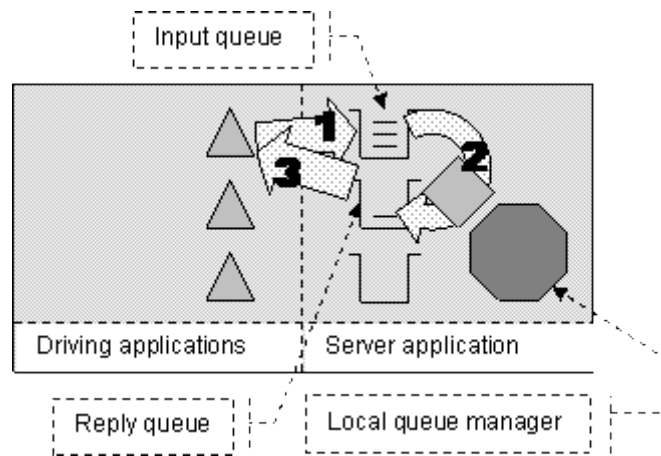


Figure 1: Connections into a local queue manager

- 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 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.

Non persistent and persistent messages were used in the local queue manager tests, with a messages size of 2K. Persistent message throughput is measured with system internal disks and an external RAID array. The effect of message throughput with large messages is investigated in section 2.1.

Figure 2 (page 3), Figure 3 (page 4) and Figure 4 (page 5) illustrate the non persistent and persistent message throughput achieved with an increasing number of driving applications in the local queue manager test scenario described in Figure 1 above observed with 16, 32, 48 and 64 server threads.

1.1.1 Non persistent messages – local queue manager

Test name – local_np

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

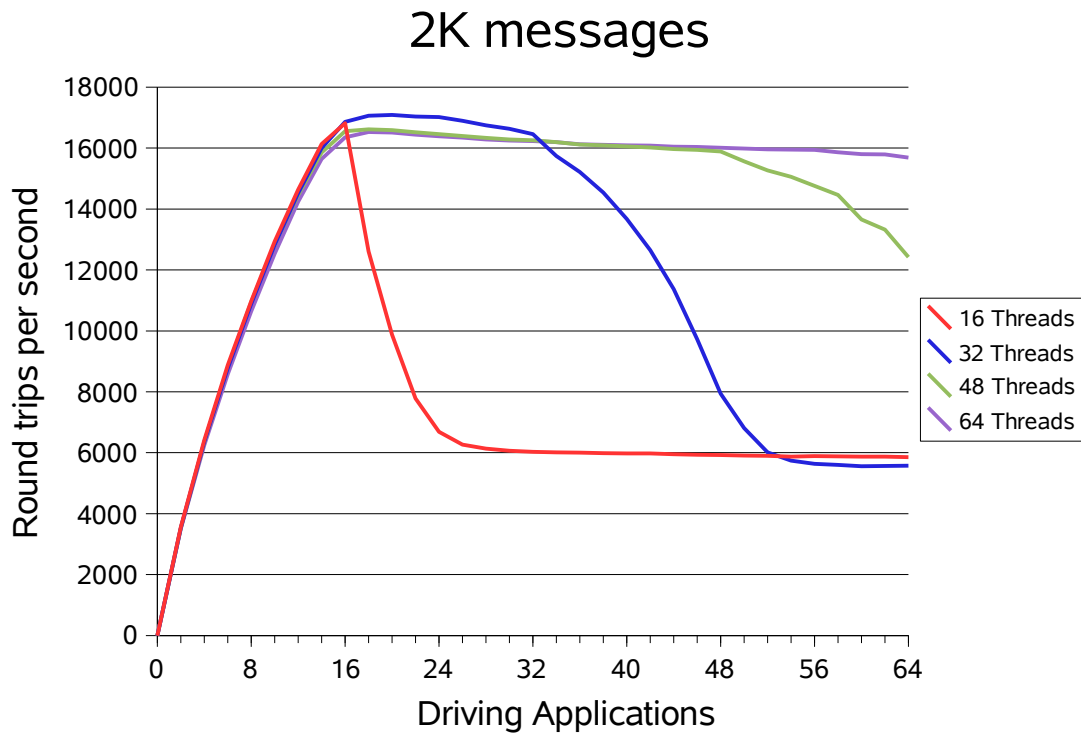


Figure 2: Non persistent 2K messages, local queue manger

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	20	17086	0.001	96%

Table 1: Peak non persistent 2K message performance, local queue manager

1.1.2 Persistent messages – local queue manager – internal disks

Test name - local_pm

Queue Manager Configuration

LogPrimaryFiles = 8, LogFilePages = 16384, LogBufferPages = 512

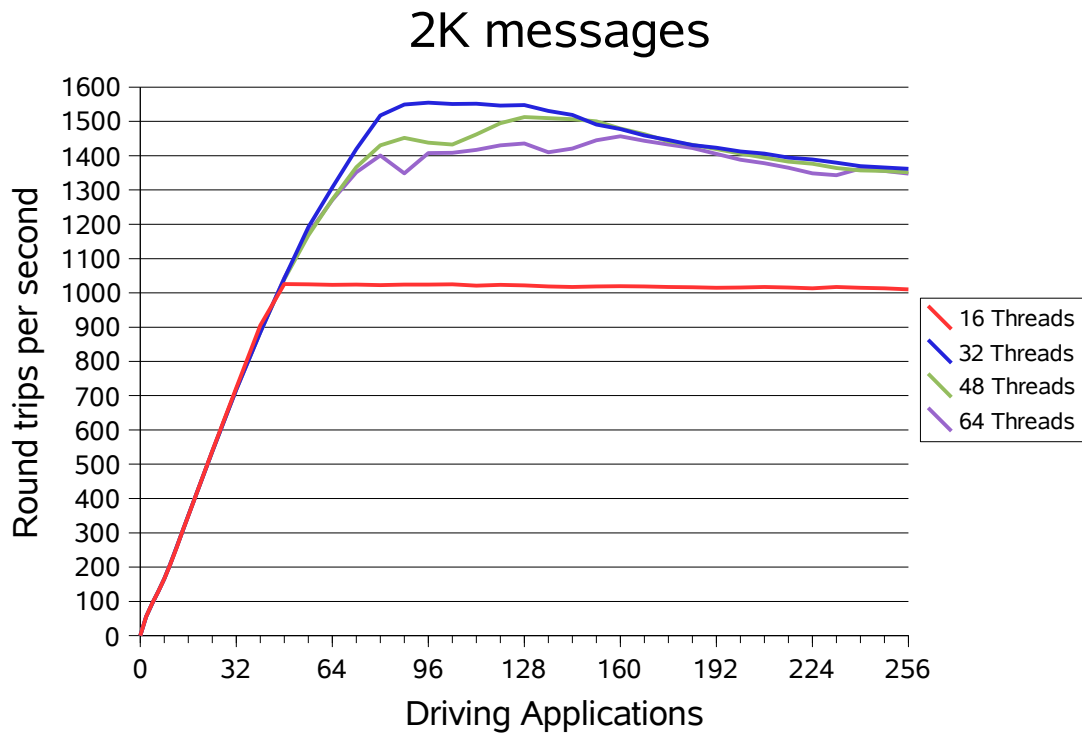


Figure 3: Persistent 2K messages, local queue manager, internal disk

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	112	1551	0.082	19%

Table 2: Peak persistent 2K message performance, local queue manager, internal disks

1.1.3 Persistent messages – local queue manager – external disks

Test name – local_pm

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

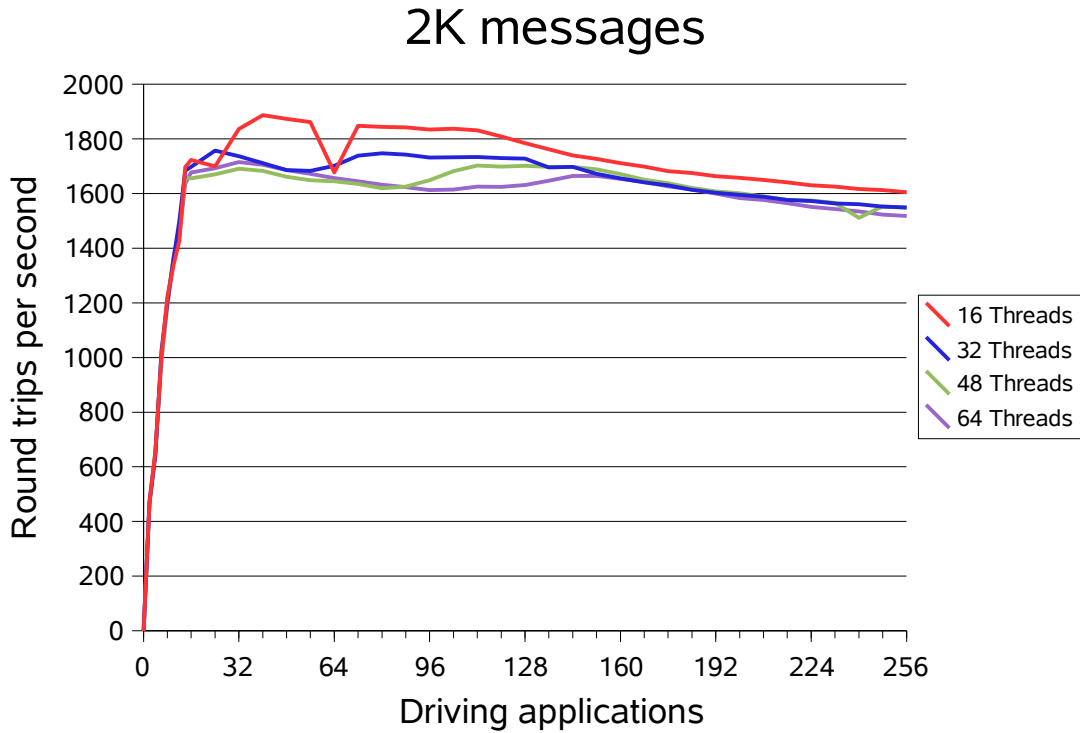


Figure 4: Persistent 2K messages, local queue manager, external disk

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	16	40	1886	0.025	21%

Table 3: Peak persistent 2K messages performance, local queue manager, external disks

1.2 Client channels test scenario between hosts

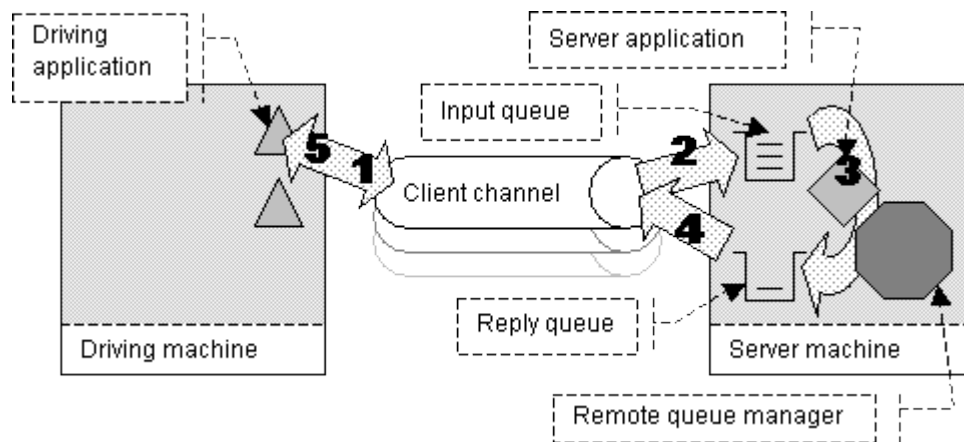


Figure 5: MQI-client channels into a remote queue manager

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 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.

Non persistent and persistent messages were used in the client channels tests, with a messages size of 2K. Persistent message throughput is measured with system internal disks and an external RAID array. The effect of message throughput with large messages is investigated in section 2.2

Figure 6 (page 7), Figure 7 (page 8) and Figure 8 (page 9) illustrate the non persistent and persistent message throughput achieved using an increasing number of driving applications in the client channel test scenario described in Figure 5 above observed with 16, 32, 48 and 64 server threads

1.2.1 Non persistent messages – client channels between hosts

Test name – *clnp6*

Queue Manager Configuration

LogPrimaryFiles = 3, *LogFilePages* = 64, *LogBufferPages* = 17

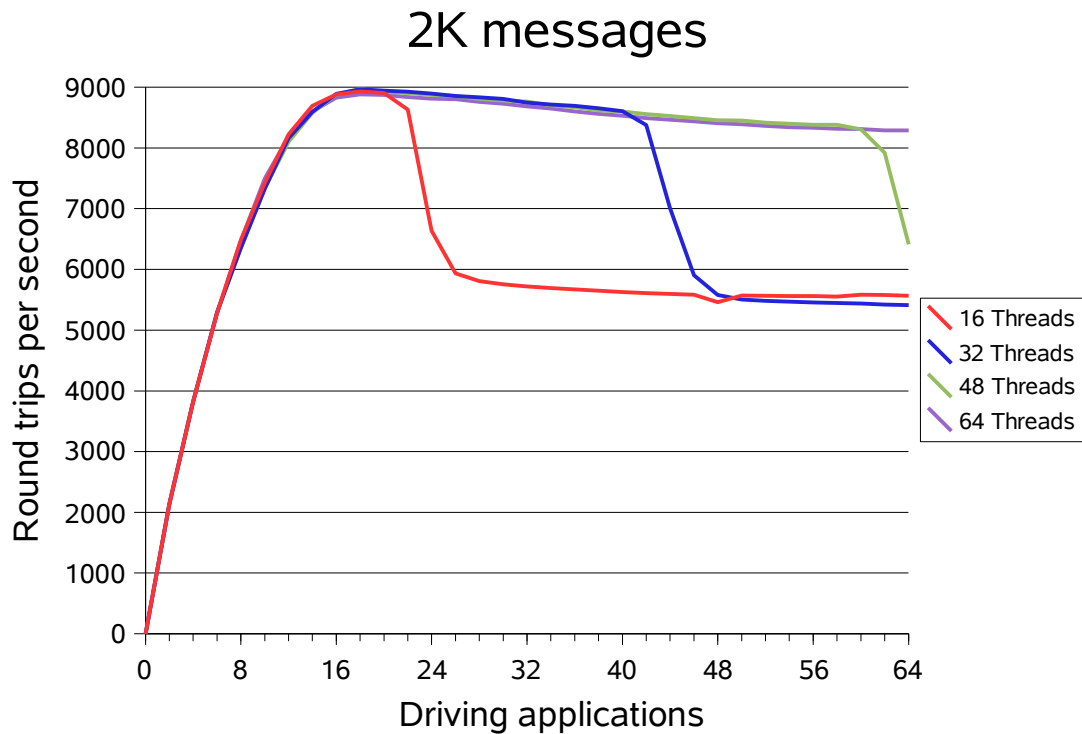


Figure 6: Non persistent 2K messages, client channels between hosts

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	18	8964	0.002	48%

Table 4: Peak non persistent 2K messages, client channels between hosts

1.2.2 Persistent messages – client channels between hosts– internal disks

Test name *clpm6*

Queue Manager Configuration

LogPrimaryFiles = 8, LogFilePages = 16384, LogBufferPages = 512

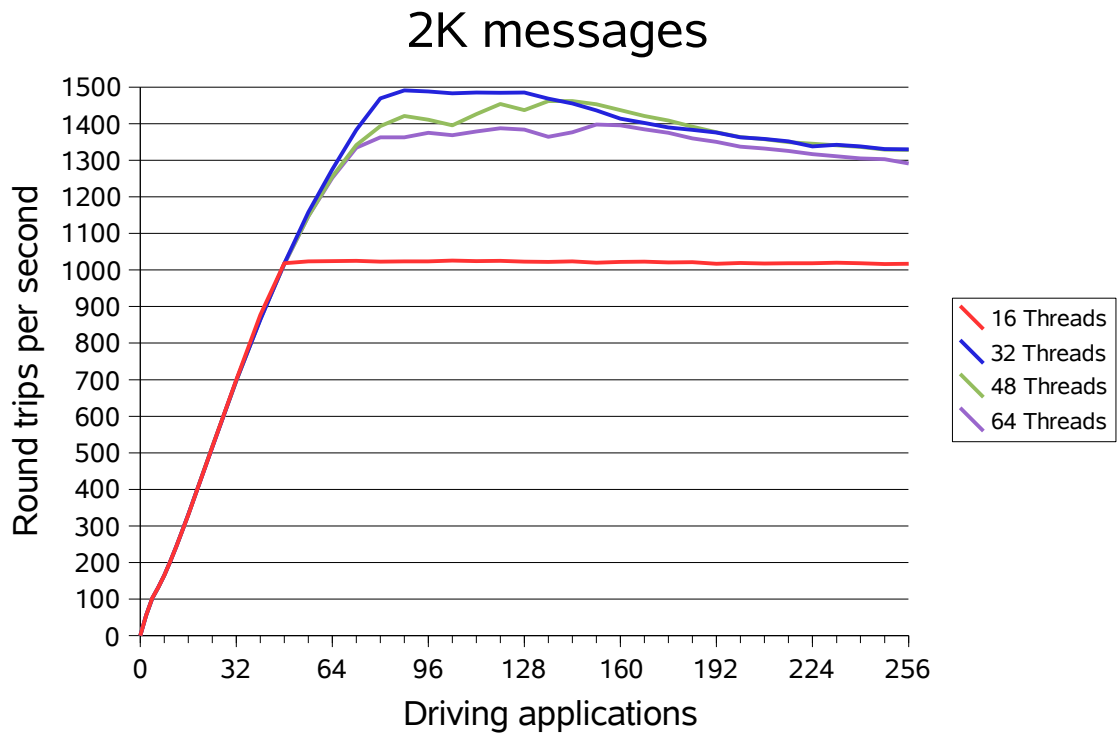


Figure 7: Persistent 2K messages, client channels between hosts, internal disks

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	88	1491	0.066	18%

Table 5: Peak persistent 2K message performance, client channels between hosts, internal disks

1.2.3 Persistent messages – client channels between hosts – external disks

Test name – *clpm6*

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

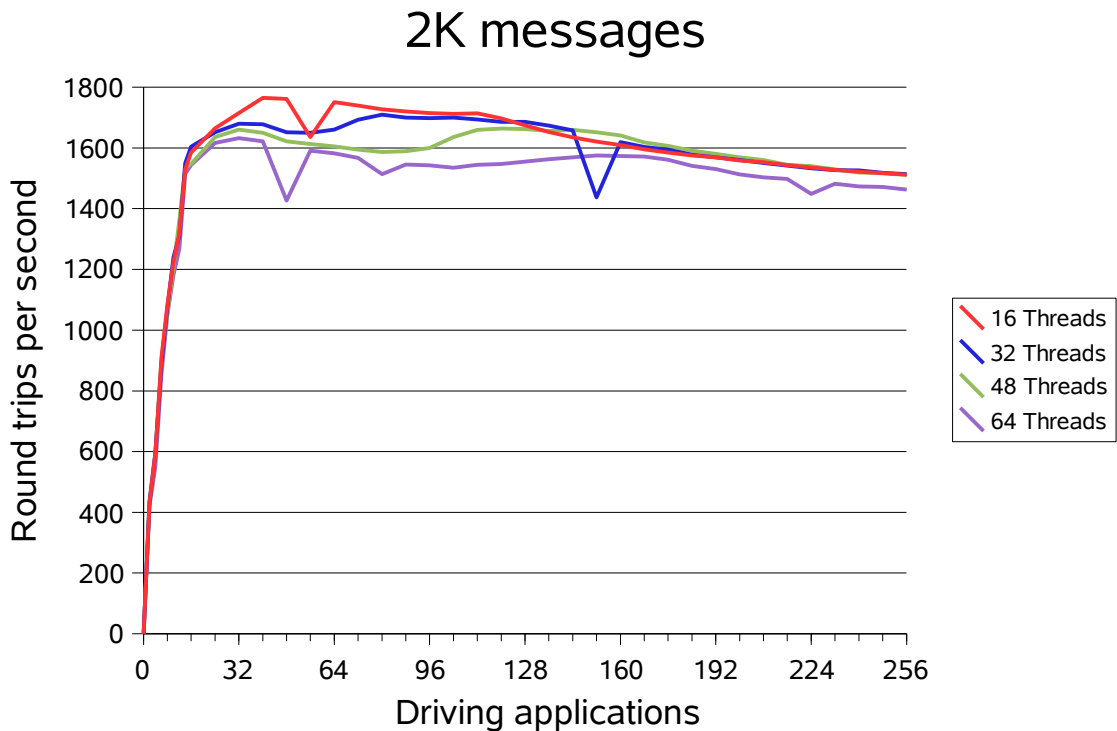


Figure 8: Persistent 2K messages, client channels between hosts, external disks

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	16	40	1765	0.026	21%

Table 6: Peak persistent 2K message performance, client channels between hosts, external disks

1.3 Client channels test scenario between zones

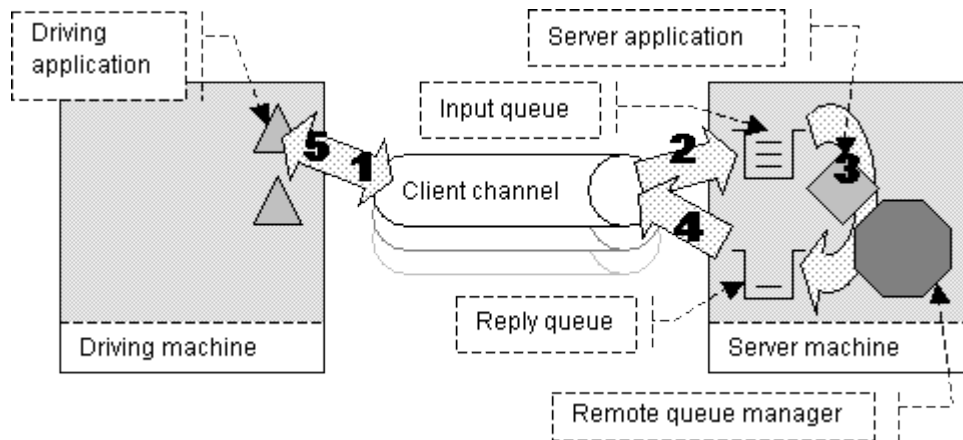


Figure 9: MQI client channels into a remote queue manager using zones

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 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.

This scenario is same as the client channels between hosts scenario in section 1.2 except that the driver and server are deployed in separate zones on the same host.

Non persistent and persistent messages were used in the client channel tests, with a message size of 2K. Persistent messages were only measured using an external RAID array. The effect of message throughput with larger messages sizes is investigated in section 2.3.

Figure 10 (page 11) and Figure 11 (page 13) illustrate the non persistent and persistent message throughput achieved using an increasing number of driving applications in the client channel test scenario described in Figure 9 above observed with 16, 32, 48 and 64 server threads.

1.3.1 Non persistent messages – client channels between zones

Test name – *clnp6*

Queue Manager Configuration

LogPrimaryFiles = 3, *LogFilePages* = 64, *LogBufferPages* = 17

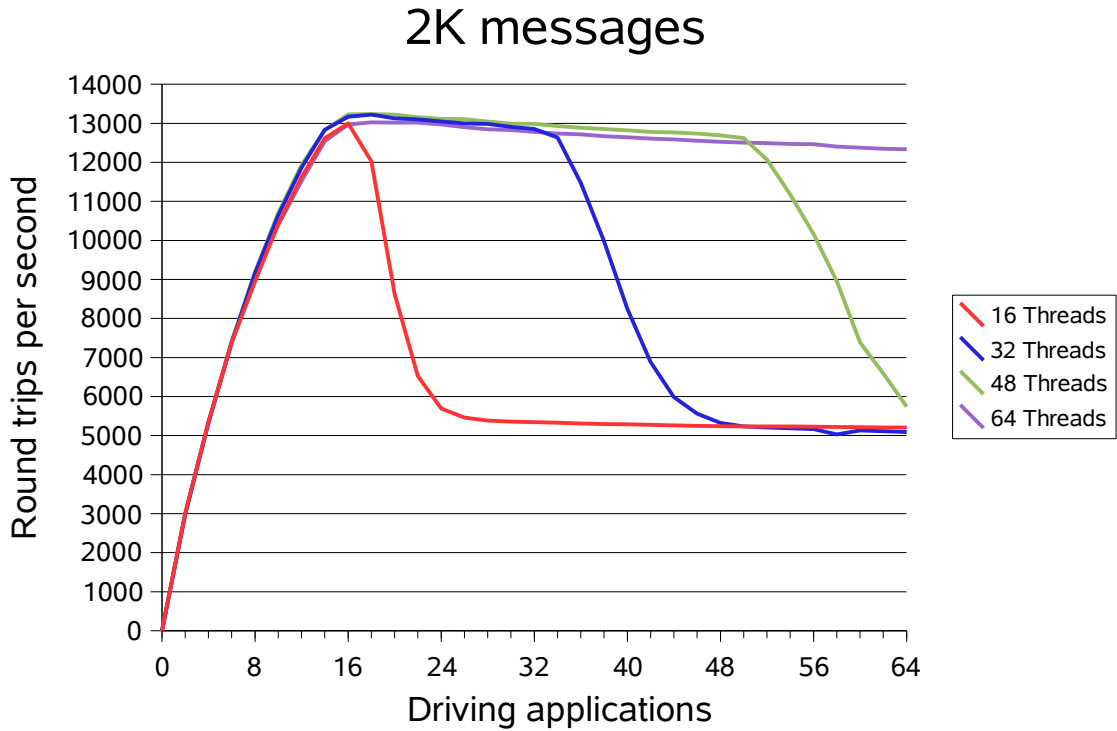


Figure 10: Non persistent 2K messages, client channels between zones

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	18	13221	0.002	93%

Table 7: Peak non persistent 2K message performance, client channels between zones

1.3.2 Persistent messages – client channels between zones – internal disks

This configuration was not tested

1.3.3 Persistent messages – client channels between zones – external disks

Test name – *clpm6*

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

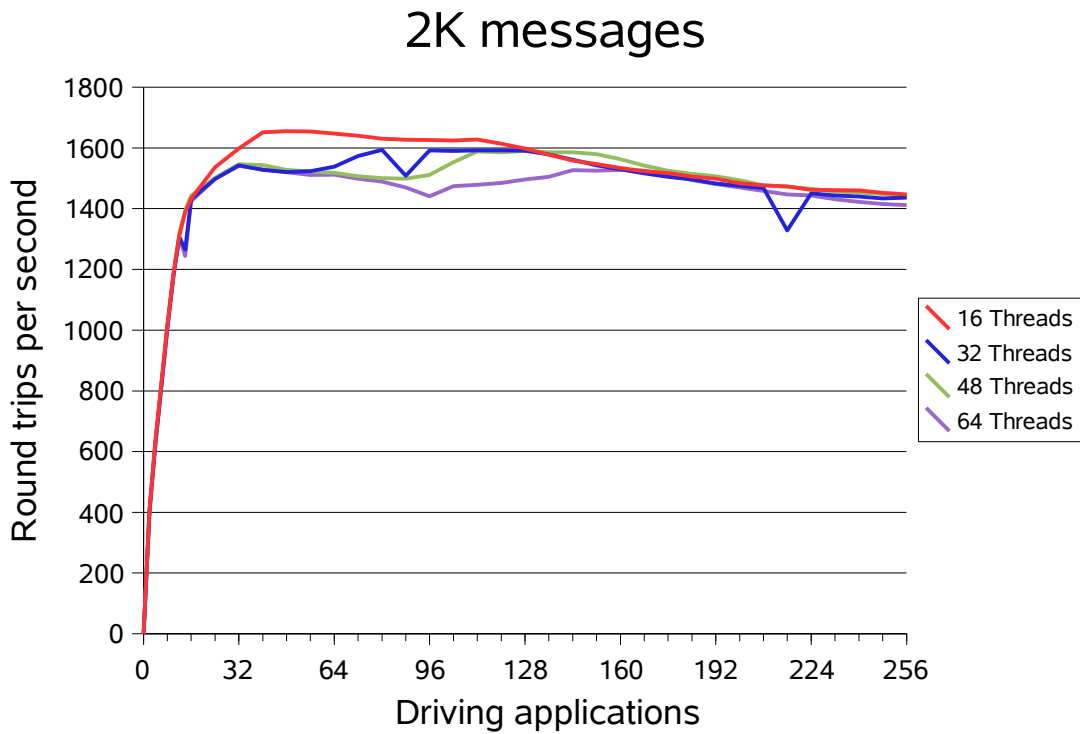


Figure 11: Persistent 2K messages, client channels between zones, external disks

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	16	48	1655	0.035	22%

Table 8: Peak persistent 2K message performance, client channels between zones, external disks

1.4 Distributed channels test scenario

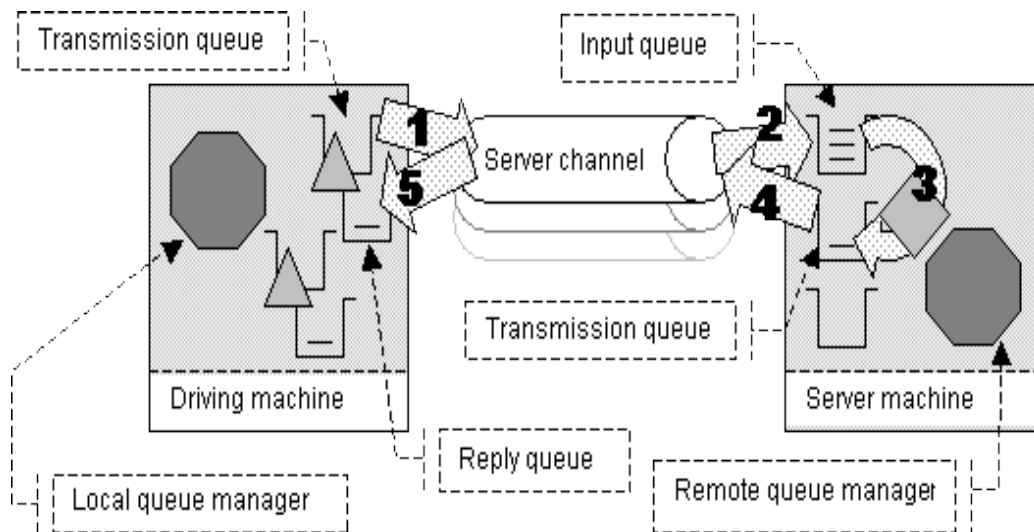


Figure 12: Server channels between two queue managers

- 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 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, 4) 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.
- 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.

Non persistent and persistent messages were used in the distributed queuing tests, with a message size of 2K. Persistent message throughput is measured with system internal disks and an external RAID array. The effect of message throughput with larger messages sizes is investigated in section 2.4

Figure 13 (page 15), Figure 14 (page 16) and Figure 15 (page 17) illustrate the non persistent and persistent message throughput achieved using an increasing number of driving applications in the server channels test scenario described in Figure 12 above observed with 16, 32, 48 and 64 server threads..

1.4.1 Non persistent messages – server channels

Test name – dqnp6

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

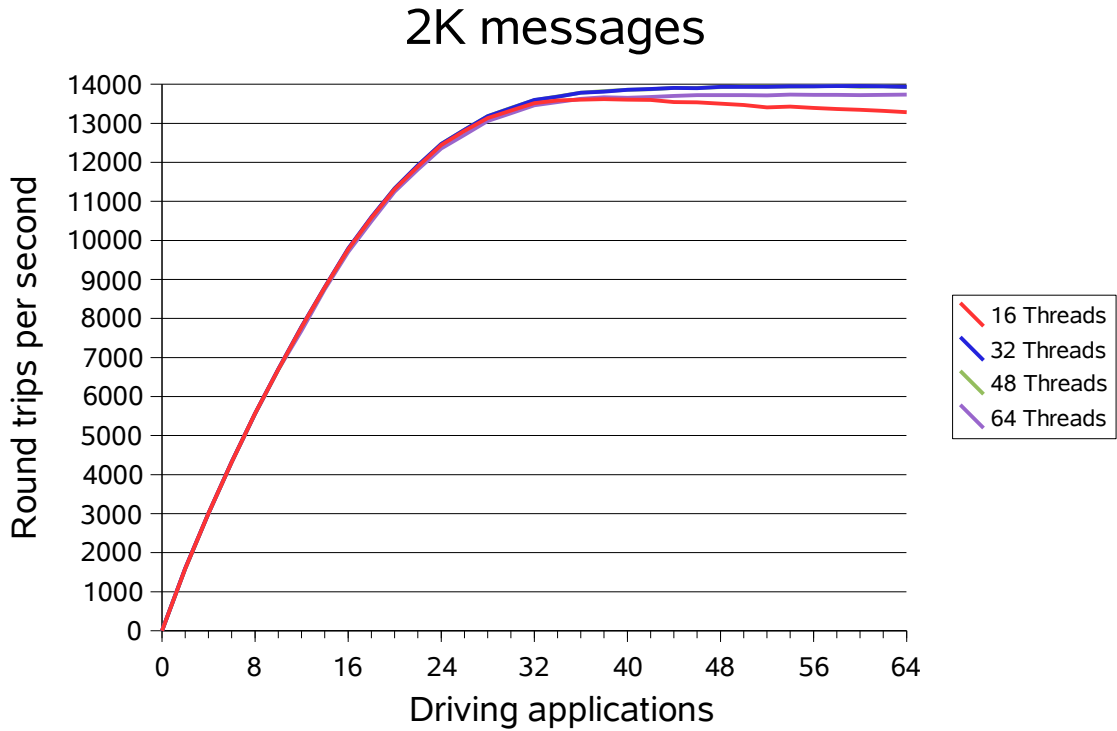


Figure 13: Non persistent 2K messages, server channels

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	54	13970	0.004	68%

Table 9: Peak non persistent 2K message performance, server channels

1.4.2 Persistent messages – server channels – internal disks

Test name – dqpm6

Queue Manager Configuration

LogPrimaryFiles = 16, LogFilePages = 16384, LogBufferPages = 512

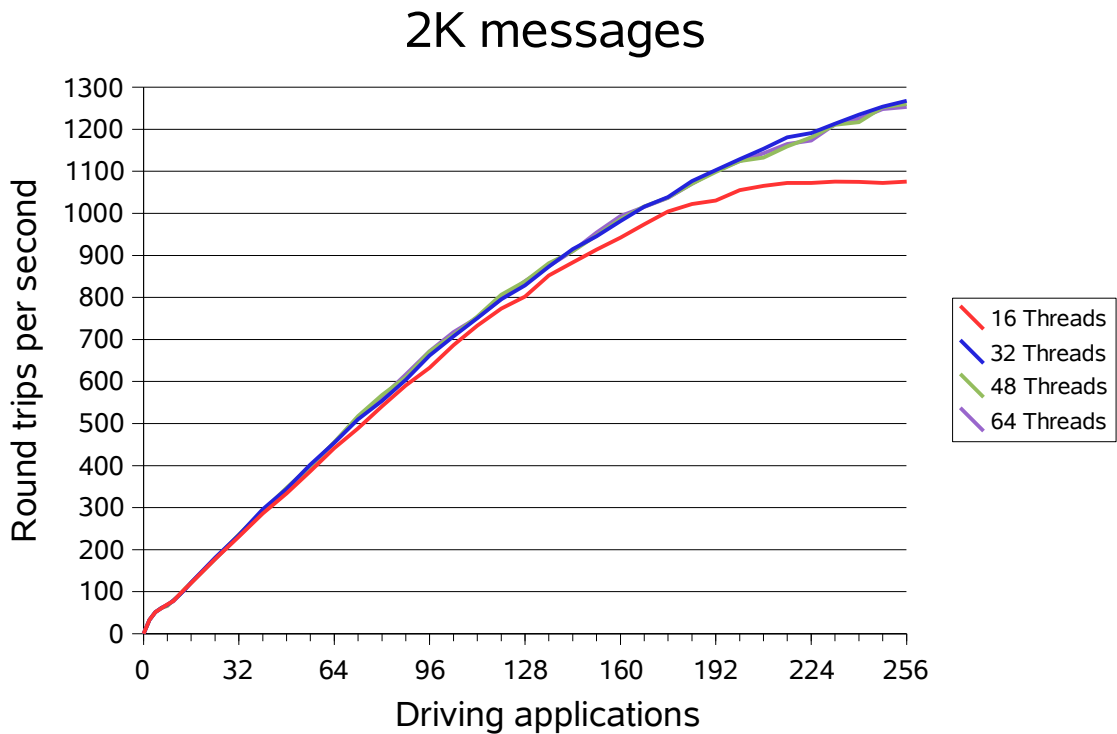


Figure 14: Persistent 2K messages, server channels, internal disks

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	32	256	1268	0.230	9%

Table 10: Peak persistent 2K message performance, server channels, internal disks

This test was terminated after 256 driving applications had connected. With 32 or more server threads the peak throughput has not been reached.

1.4.3 Persistent messages – server channels – external disks

Test name – dqpm6

Queue Manager Configuration

LogPrimaryFiles = 16, LogFilePages = 16384, LogBufferPages = 512

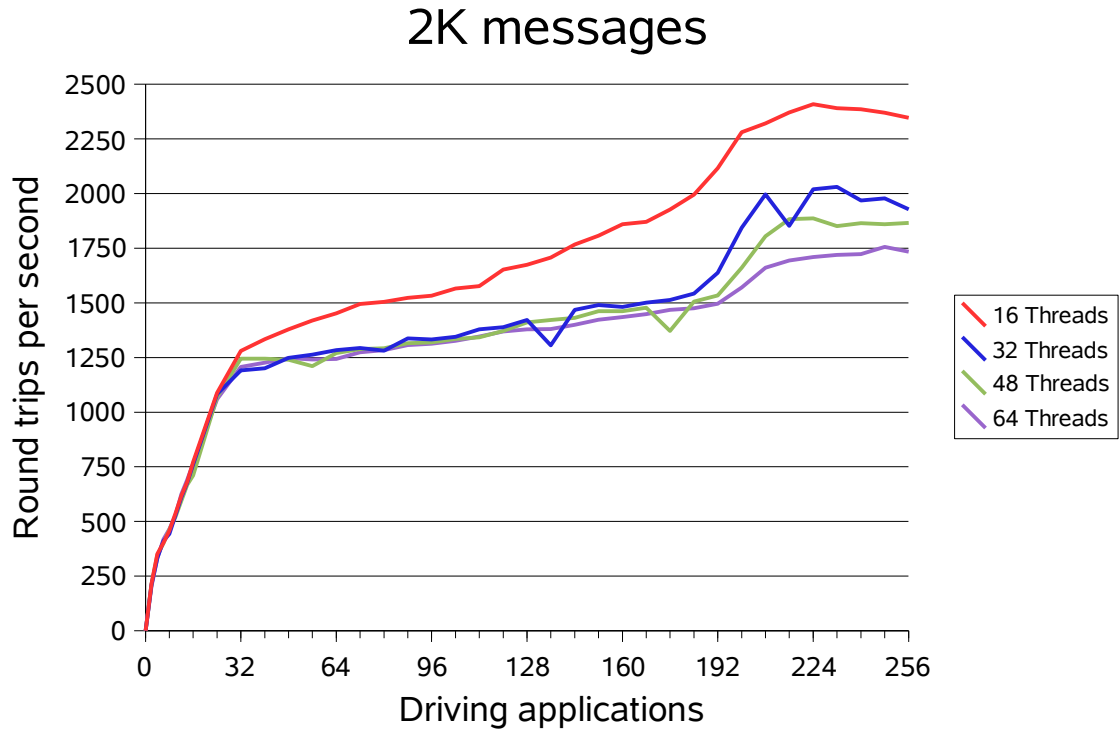


Figure 15: Persistent 2K messages, server channels, external disks

Message size	Server threads	Driving applications	Round trips per second	Response time	CPU usage
2K	16	224	2409	0.107	19%

Table 11: Peak persistent 2K message performance, server channels, external disks

2 Large Messages

2.1 Local queue manager test scenario – large messages

See section 1.1 for a description of this test scenario.

2.1.1 Non persistent large messages – local queue manager

Test name – local_np

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

The throughput of 20K and 200K non persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the non persistent local queue manager scenario are shown in Figure 16 (page 21) and Figure 17 (page 21).

Figure 18 (page 22) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 19 (page 22) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 12 below Shows the peak non persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	20	17086	0.001	96%
20K	32	20	12600	0.002	95%
200K	32	28	2545	0.013	85%

Table 12: Peak non persistent 2K, 20K & 200K message performance, local queue manager

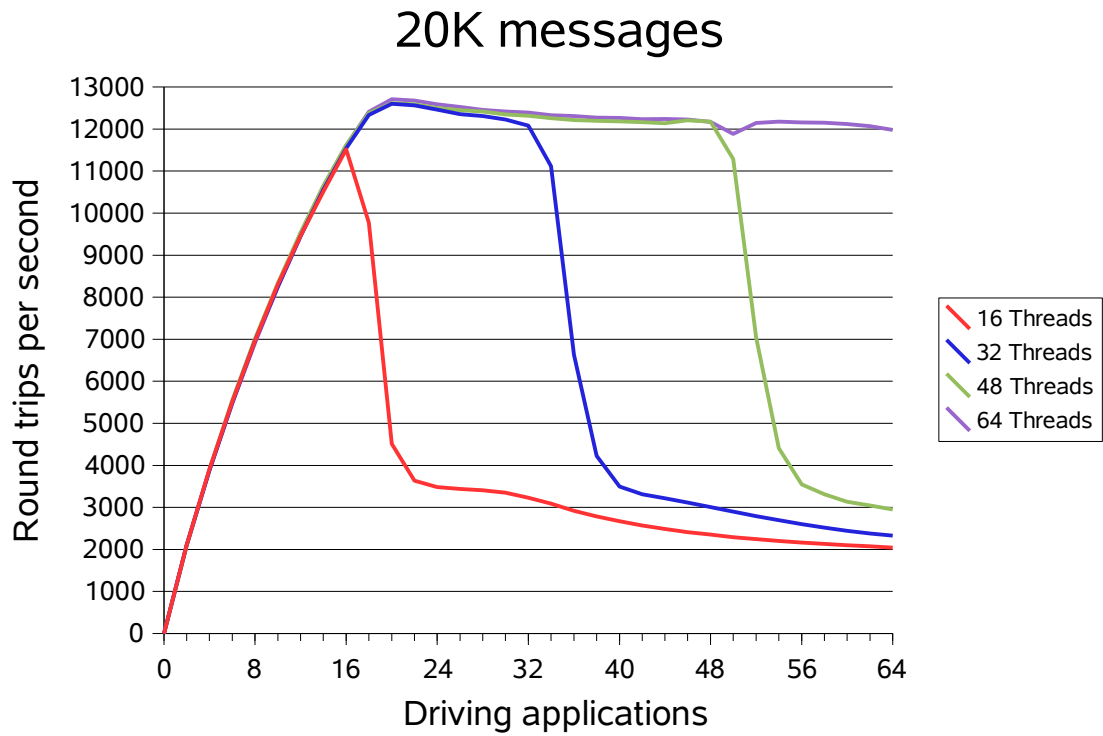


Figure 16: Non persistent 20K messages, local queue manager

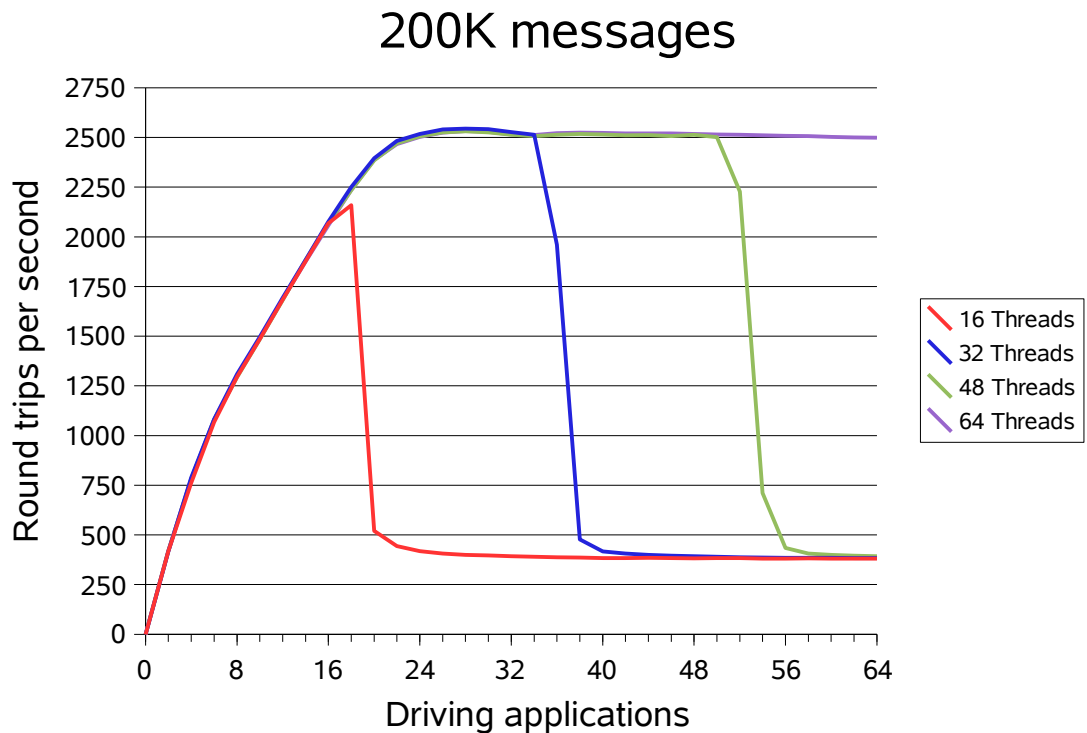


Figure 17: Non persistent 200K messages, local queue manager

2K, 20K and 200K messages

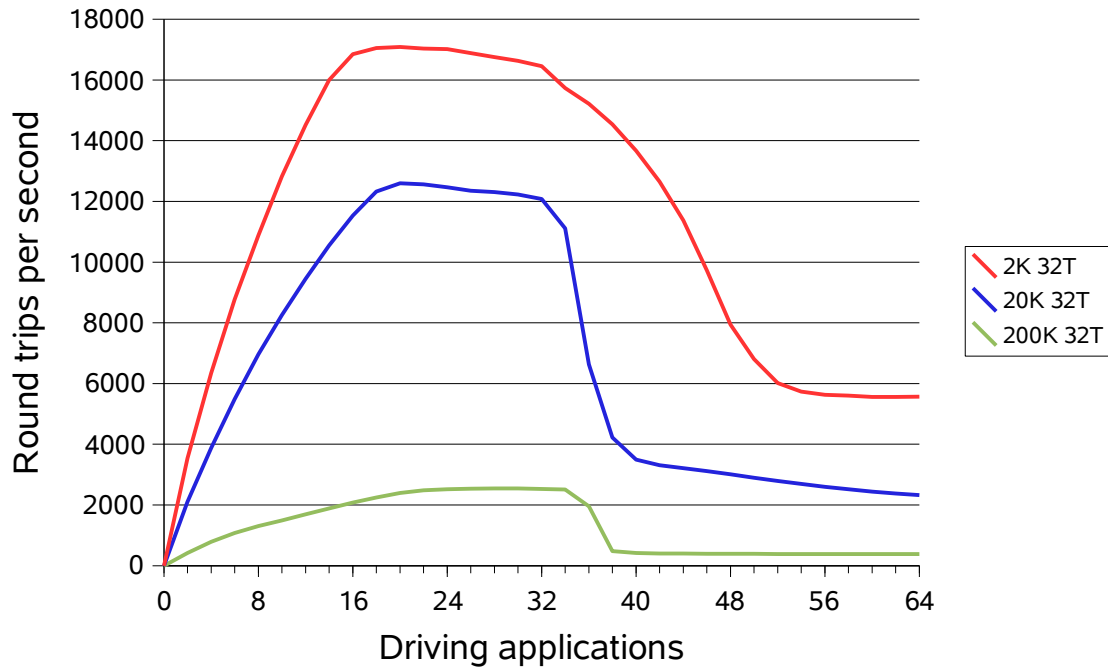


Figure 18: Comparison of non persistent 2K, 20K & 200K messages, local queue manager

1K - 256K messages

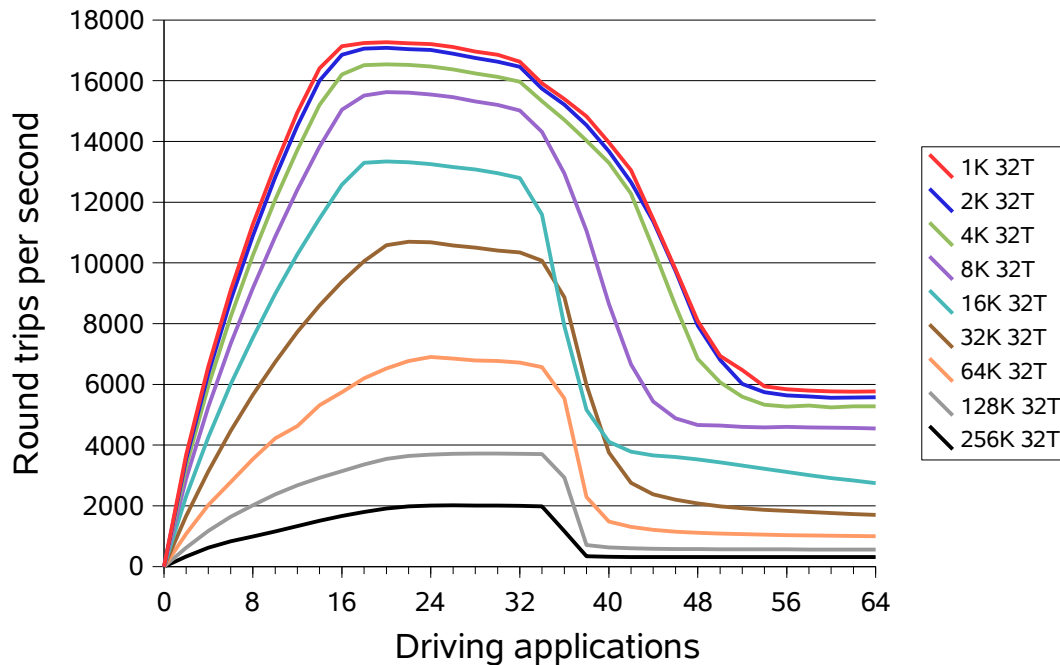


Figure 19: Comparison of non persistent messages from 1K to 256K, local queue manager

2.1.2 Persistent large messages – local queue manager – internal disks

Test name – local_pm

Queue Manager Configuration

LogPrimaryFiles = 8, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using internal disks are shown in Figure 20 (page 24) and Figure 21 (page 24).

Figure 22 (page 22) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 23 (page 25) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 13 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using internal disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	112	1551	0.082	19%
20K	16	40	288	0.159	5%
200K	16	10	33	0.352	2%

Table 13: Peak persistent 2K, 20K & 200K message performance, local queue manager, internal disks

20K messages

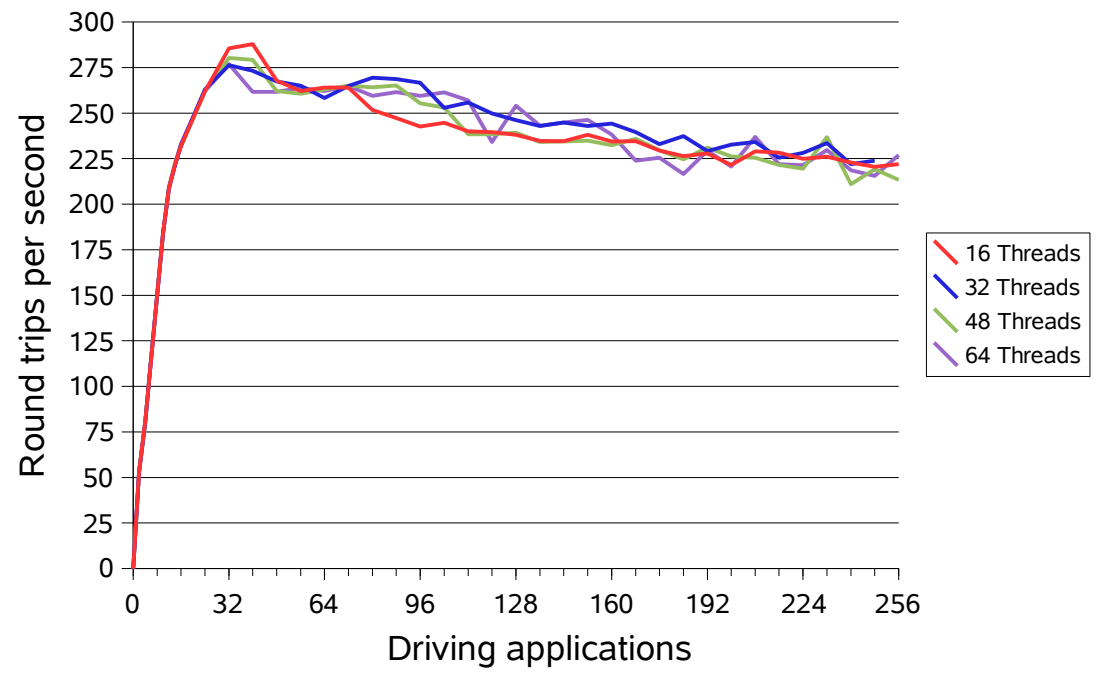


Figure 20: Persistent 20K messages, local queue manager, internal disks

200K messages

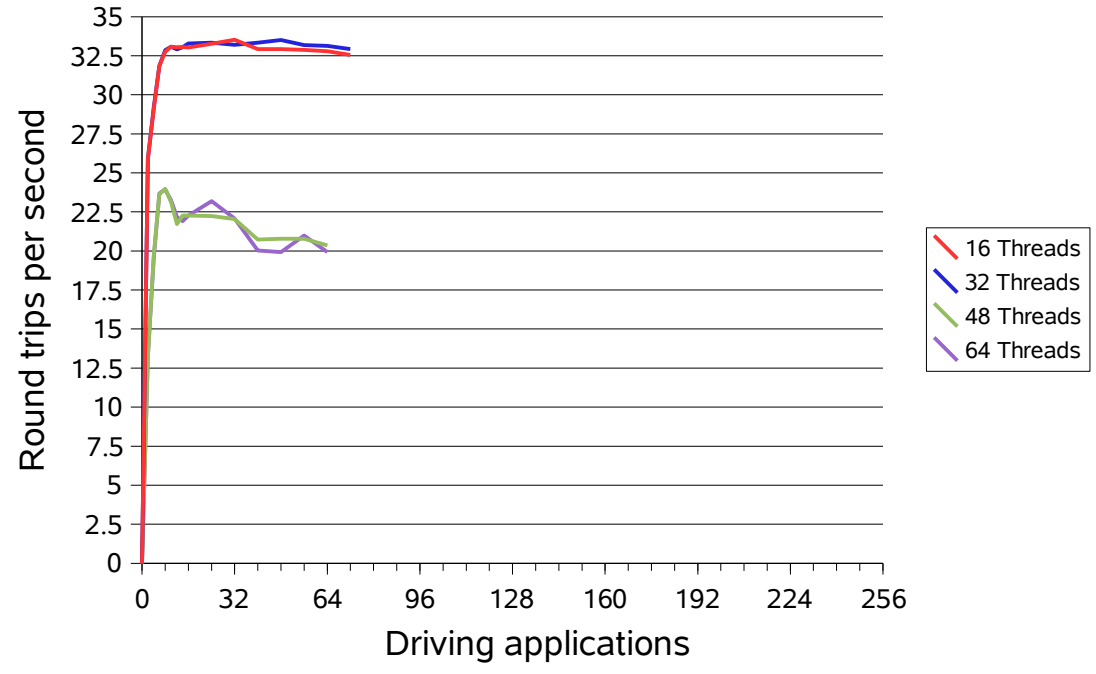


Figure 21: Persistent 200K messages, local queue manager, internal disks

2K, 20K & 200K messages

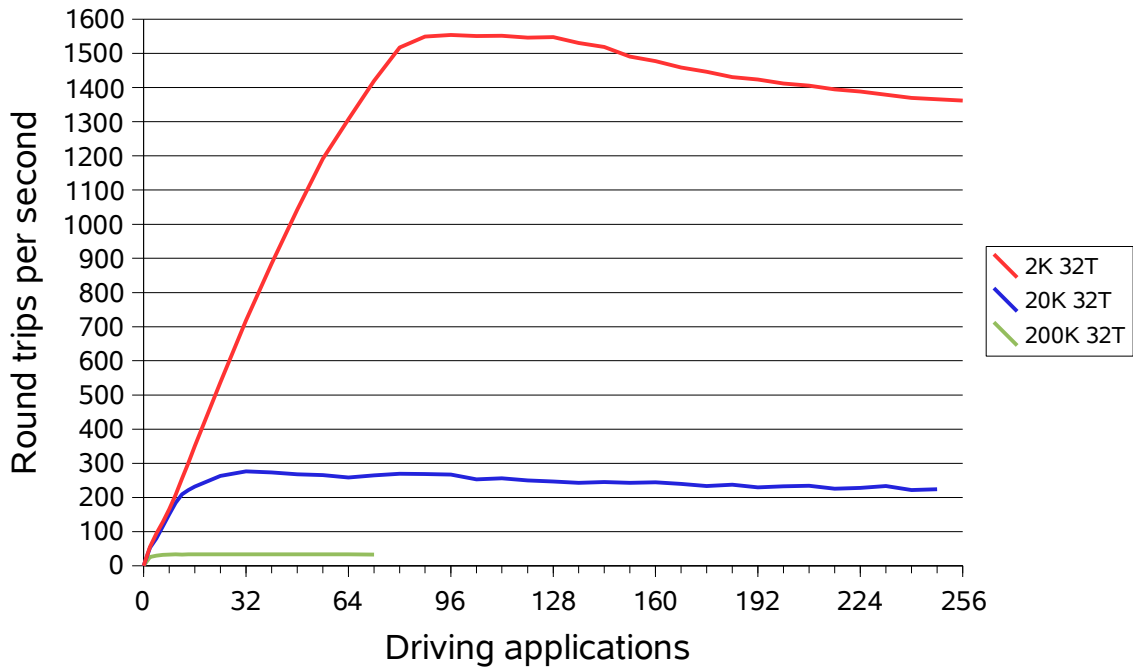


Figure 22: Comparison of 2, 20K and 200K persistent messages, local queue manager, internal disks

1K - 256K messages

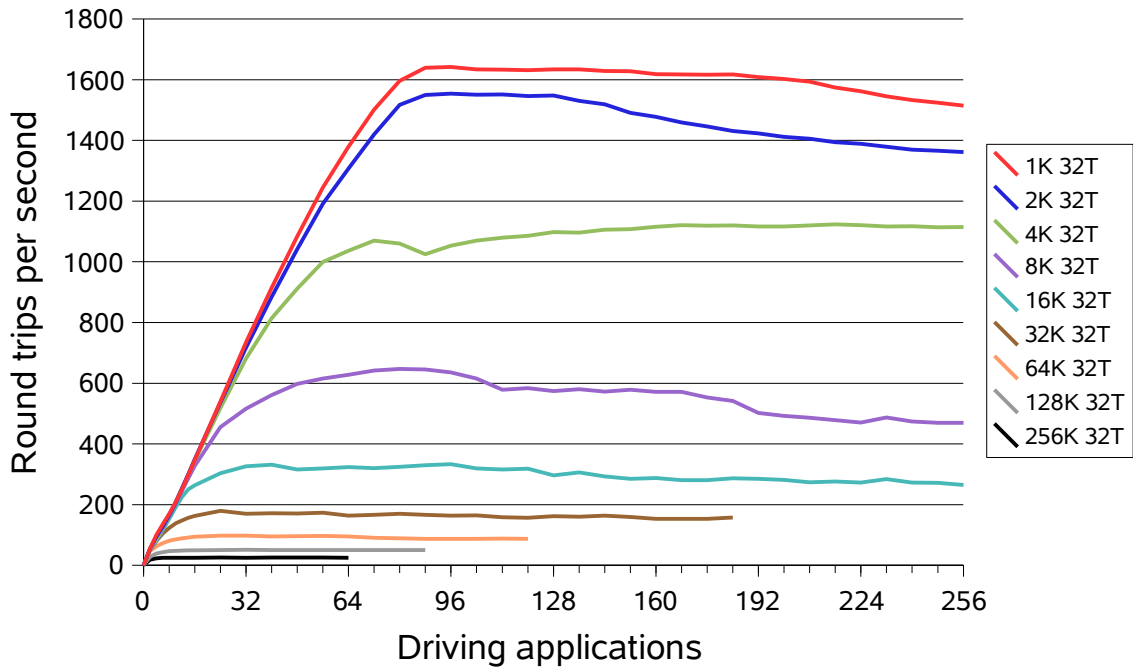


Figure 23: Comparisons of persistent messages from 1K to 256K, local queue manager, internal disks

2.1.3 Persistent large messages – local queue manager – external disks

Test name – local_pm

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using external disks are shown in Figure 24(page 27) and Figure 25 (page 27).

Figure 26(page 28) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 27 (page 28) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 14 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU Usage</i>
2K	16	40	1886	0.025	21%
20K	32	16	843	0.022	13%
200K	64	64	91	0.888	4%

Table 14: Peak persistent 2K, 20K & 200K message performance, local queue manager, internal disks

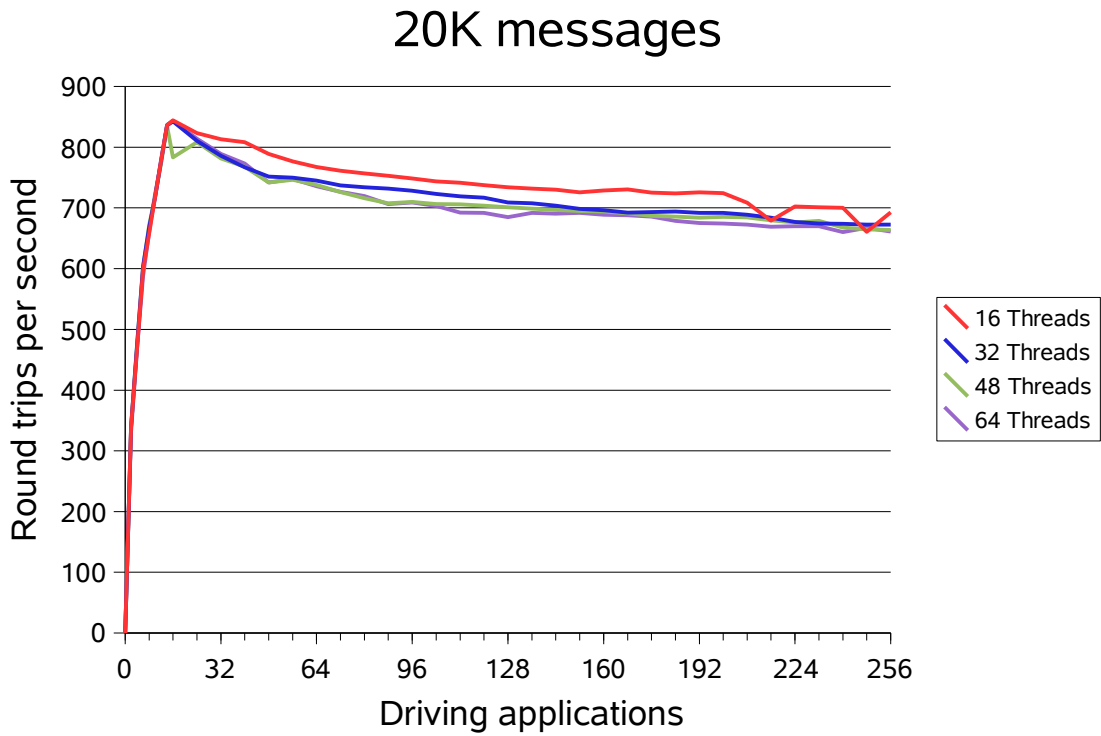


Figure 24: Persistent 20K messages, local queue manager, external disks

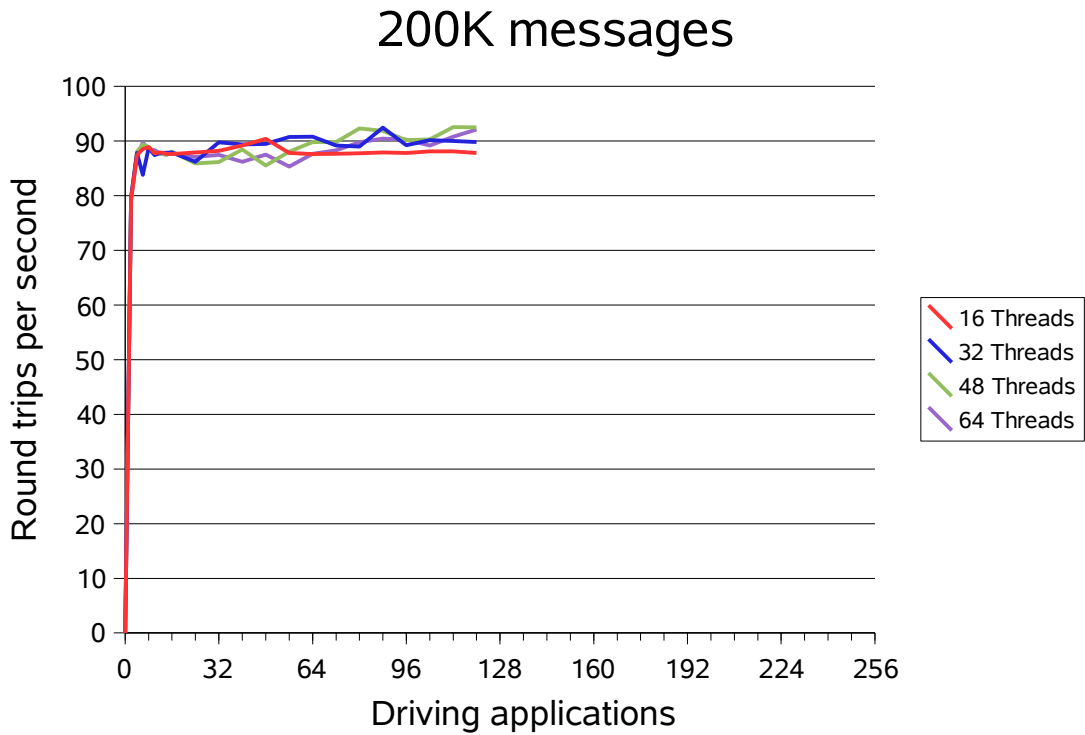


Figure 25: Persistent 200K messages, local queue manager, external disks

2K, 20K & 200K messages

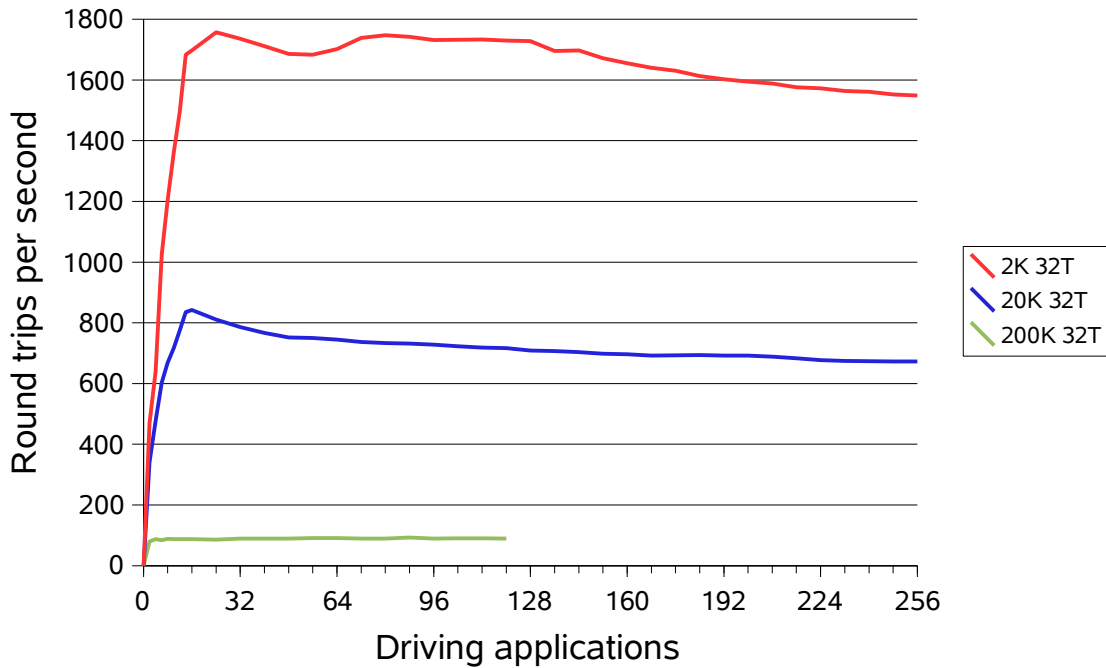


Figure 26: Comparison of 2K, 20K & 200K persistent messages, local queue manager, external disks

1K - 256K messages

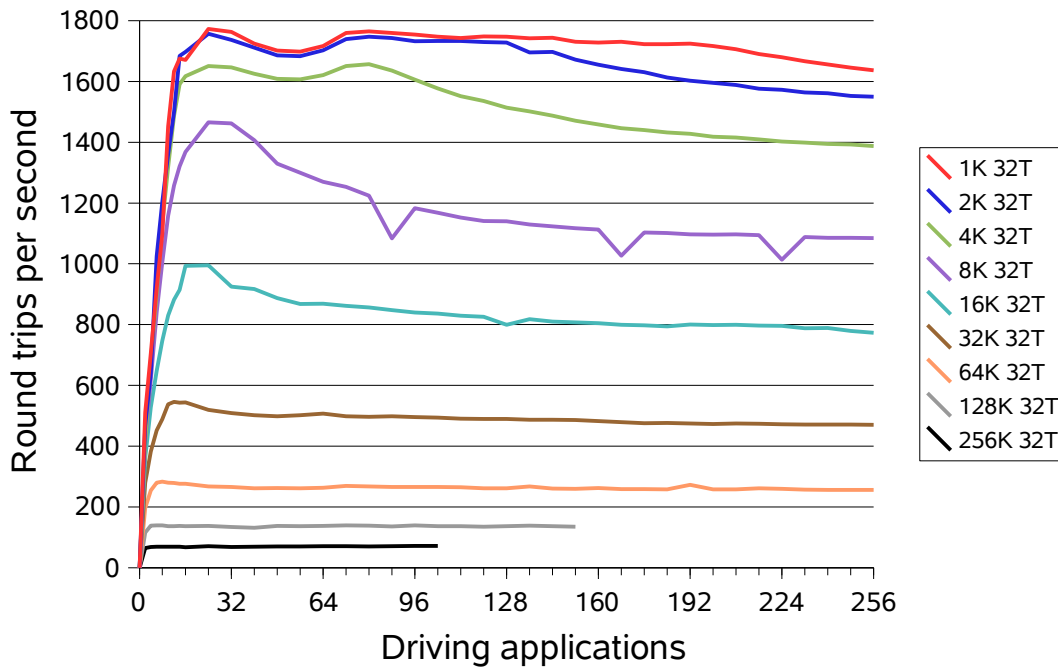


Figure 27: Comparison of persistent messages from 1K to 256K, local queue manager, external disks

2.2 Client channels test scenario between hosts

See section 1.2 for a description of this scenario.

2.2.1 Non persistent large messages – client channels between hosts

Test name – clnp6

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the non persistent client channels between hosts scenario are shown in Figure 28(page 31) and Figure 29 (page 31).

Figure 30 (page 32) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 31(page 32) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 15 below shows the peak non persistent message throughput observed for 2K, 20K and 200K messages in the client channels between hosts scenario.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	18	8964	0.002	48%
20K	32	10	3221	0.003	26%
200K	32	8	314	0.030	15%

Table 15: Peak non persistent 2K, 20K & 200K message performance, client channels between hosts

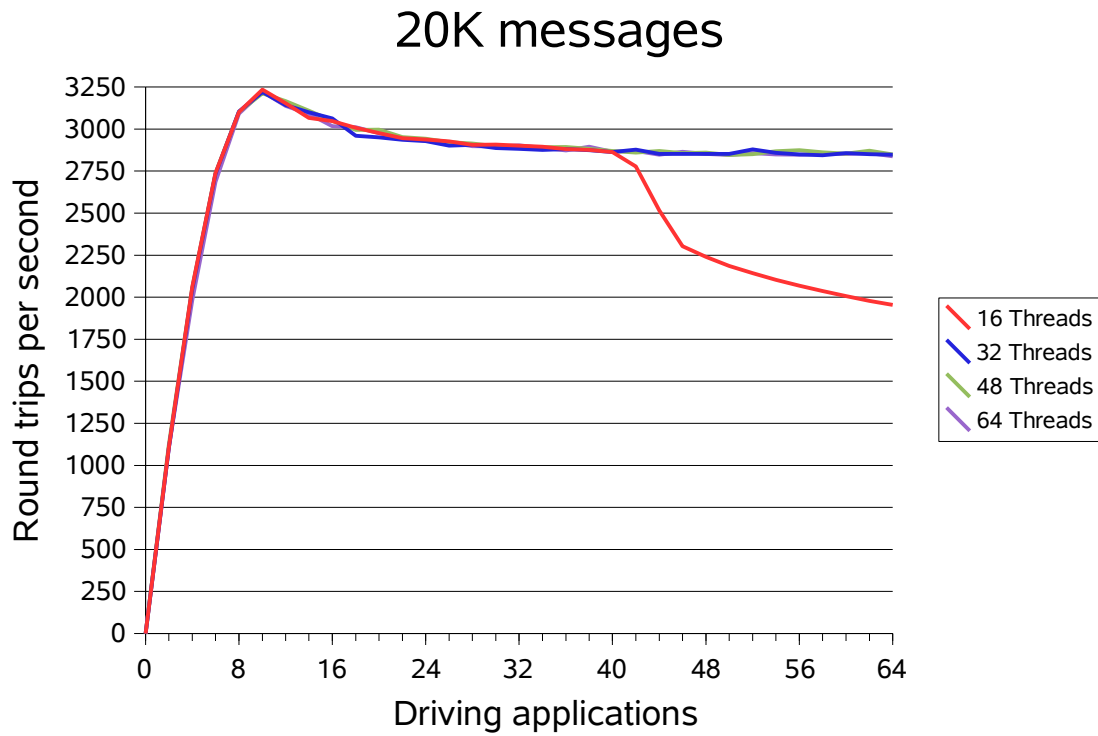


Figure 28: Non persistent 20K messages, client channels between hosts

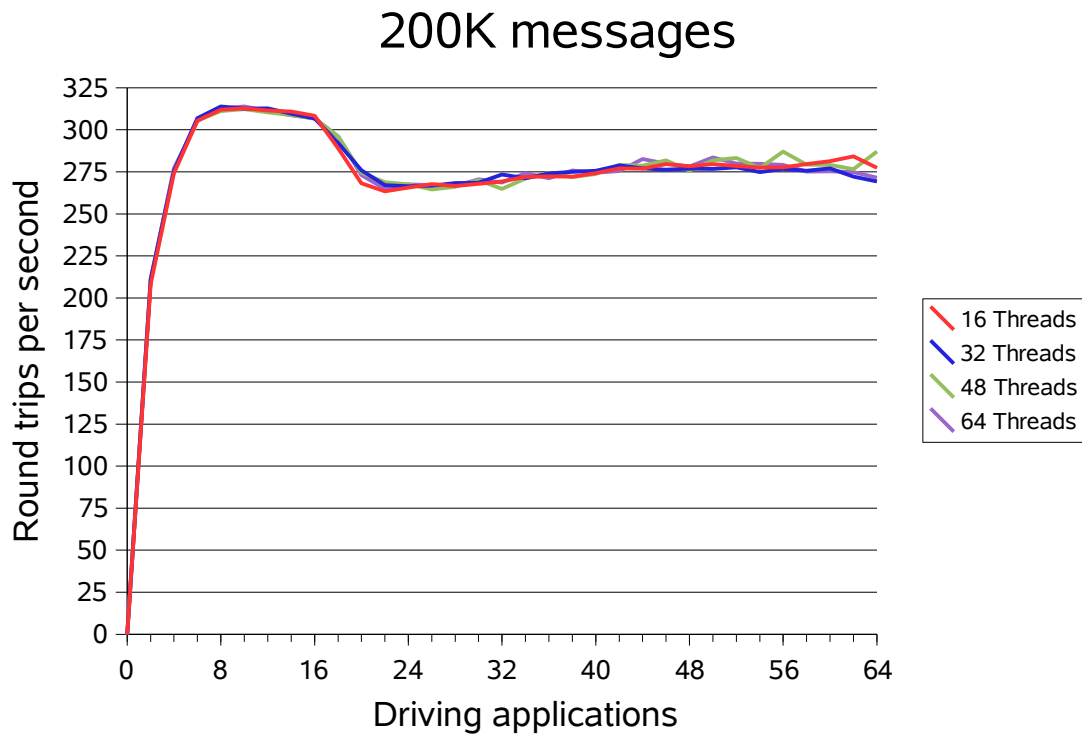


Figure 29: Non persistent 200K messages, client channels between hosts

2K, 20K & 200K messages

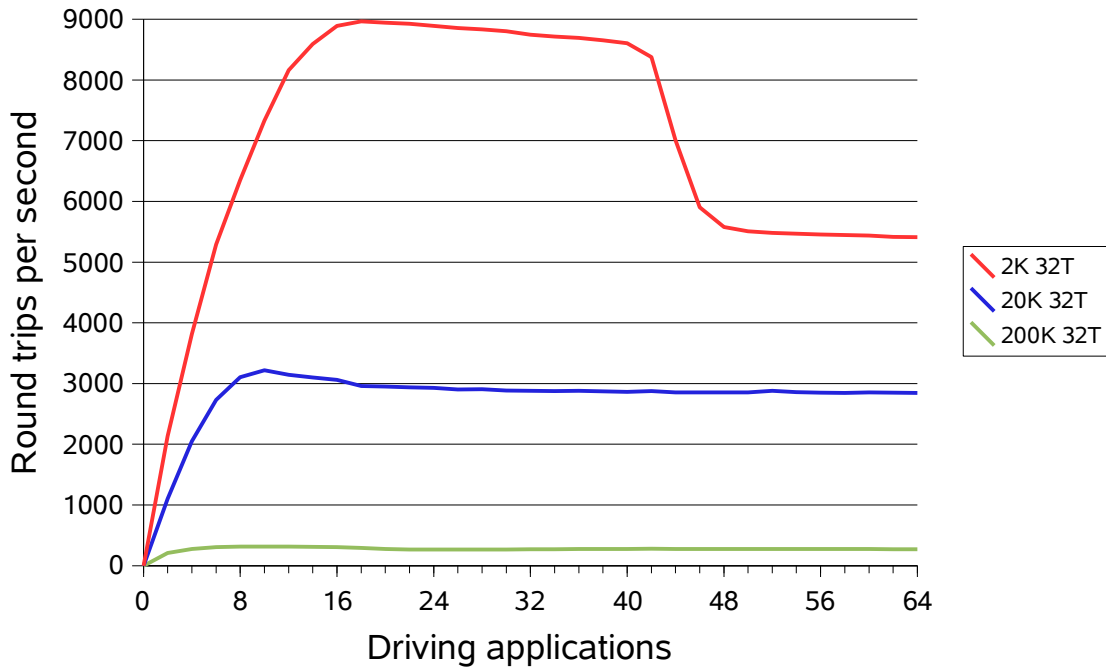


Figure 30: Comparison of 2K, 20K & 200K non persistent messages, client channels between hosts

1K - 256K messages

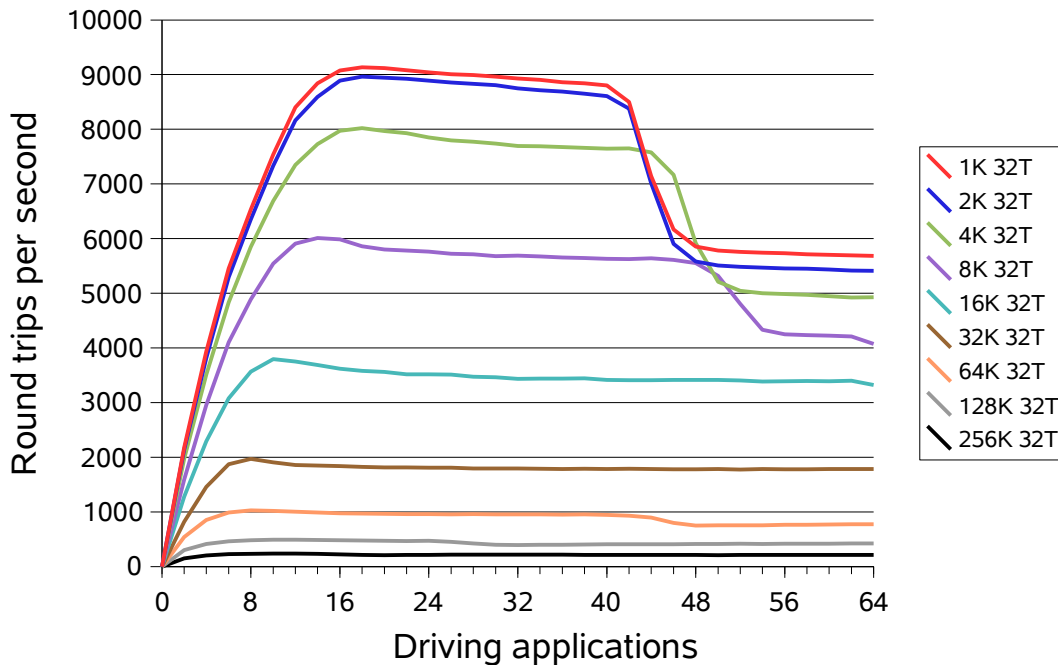


Figure 31: Comparison of non persistent messages from 1K - 256K, client channels between hosts

2.2.2 Persistent large messages – client channels between hosts– internal disks

Test name – clnp6

Queue Manager Configuration

LogPrimaryFiles = 8, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent client channels between hosts scenario using internal disks are shown in Figure 32 (page 34) and Figure 33 (page 34).

Figure 34 (page 35) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 35 (page 35) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 16 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the client channels between hosts scenario using internal disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	88	1491	0.066	18%
20K	16	32	284	0.131	5%
200K	16	24	33	0.812	3%

Table 16: Peak persistent 2K, 20K & 200K message performance, client channels between hosts, internal disks

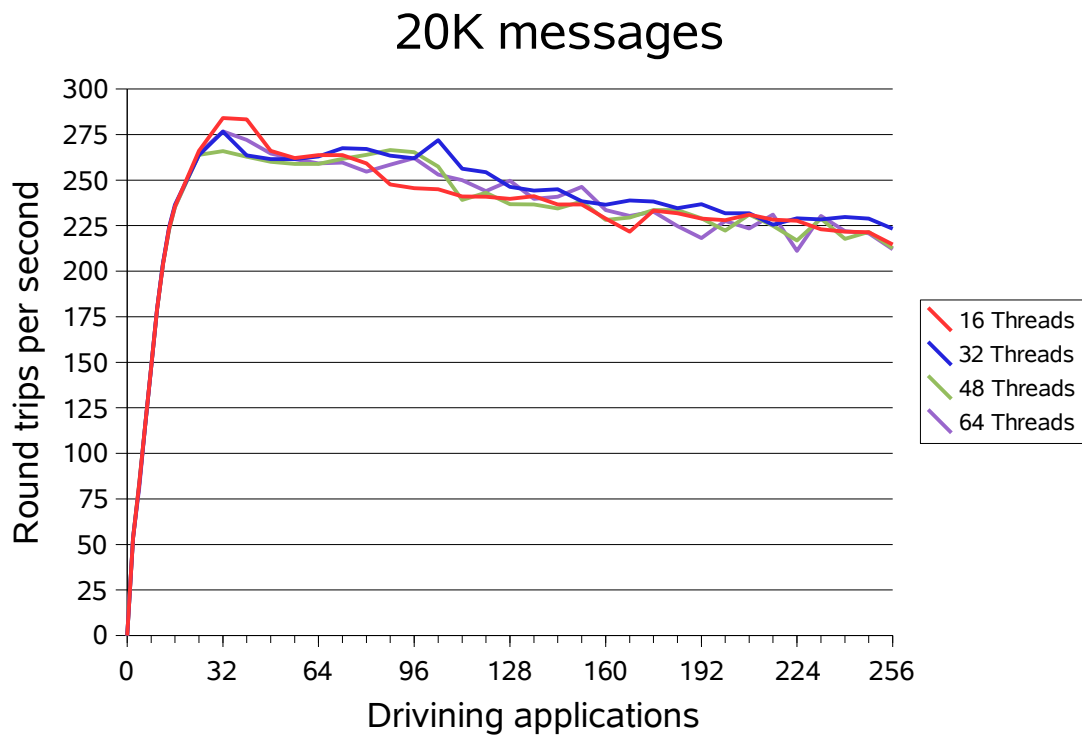


Figure 32: Persistent 20K messages, client channels between hosts, internal disks

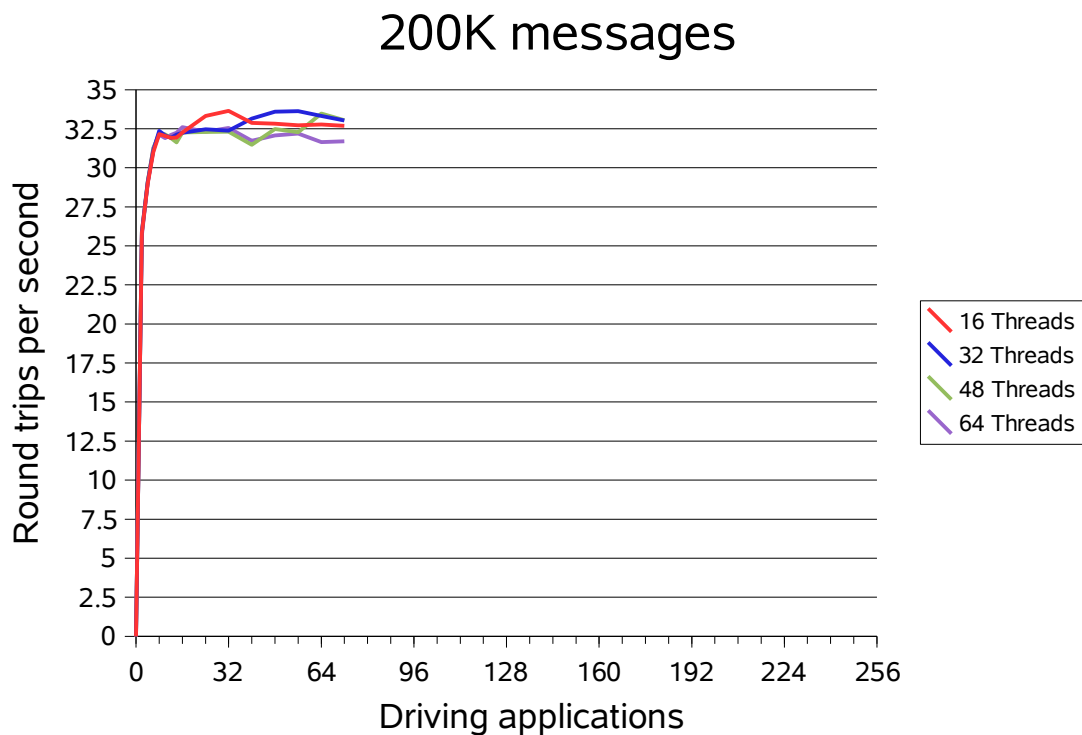


Figure 33: Persistent 200K messages, client channels between hosts, internal disks

2K, 20K & 200K messages

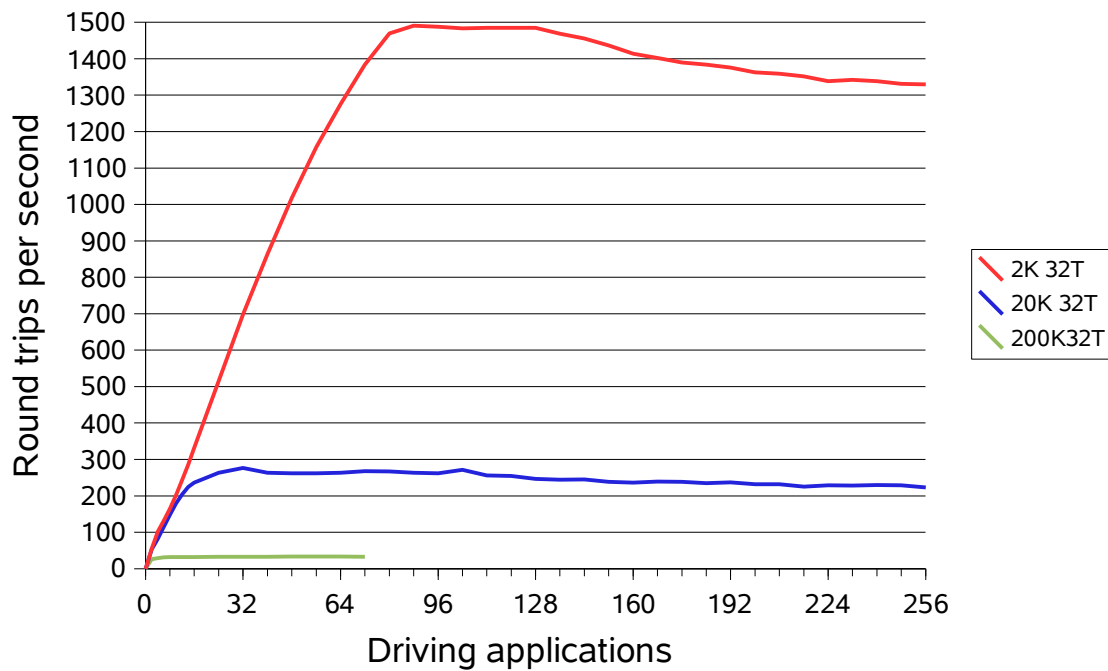


Figure 34: Comparison of 2K, 20K & 200K persistent messages, client channels between hosts, internal disks

1K - 256K messages

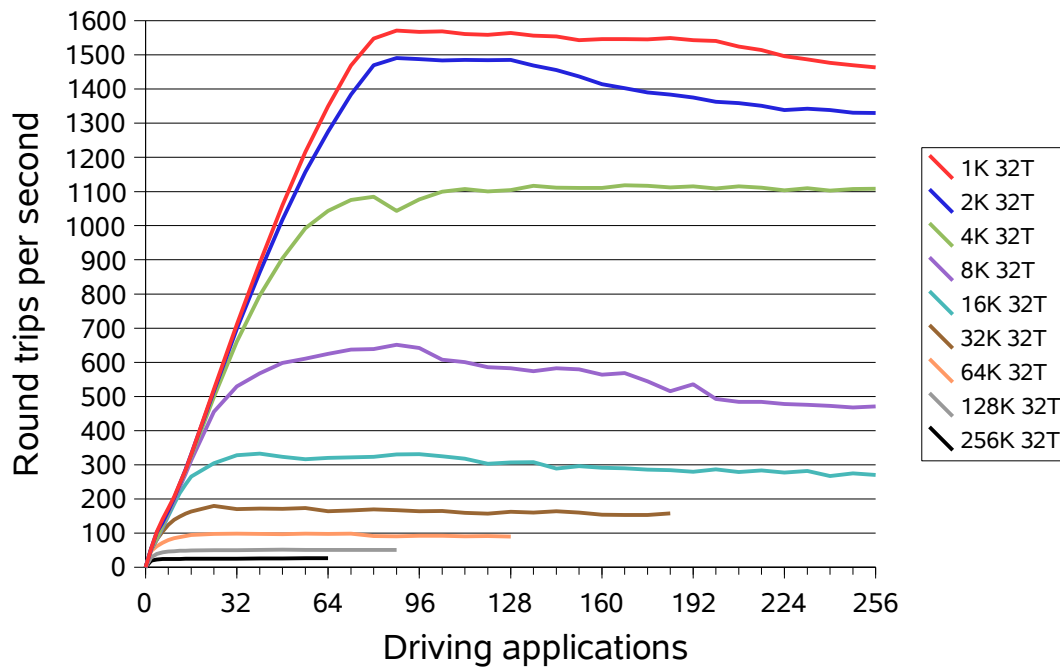


Figure 35: Comparison of persistent messages from 1K to 256K, client channels between hosts, internal disks

2.2.3 Persistent large messages – client channels between hosts – external disks

Test name – clpm6

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent client channels between hosts scenario using external disks are shown in Figure 36 (page 37) and Figure 37 (page 37).

Figure 38 (page 38) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 39 (page 38) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 17 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages client channels between hosts scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	16	40	1765	0.026	21%
20K	16	16	808	0.023	13%
200K	16	48	91	0.639	8%

Table 17: Peak persistent 2K, 20K & 200K message performance, client channels between hosts, external disks

20K messages

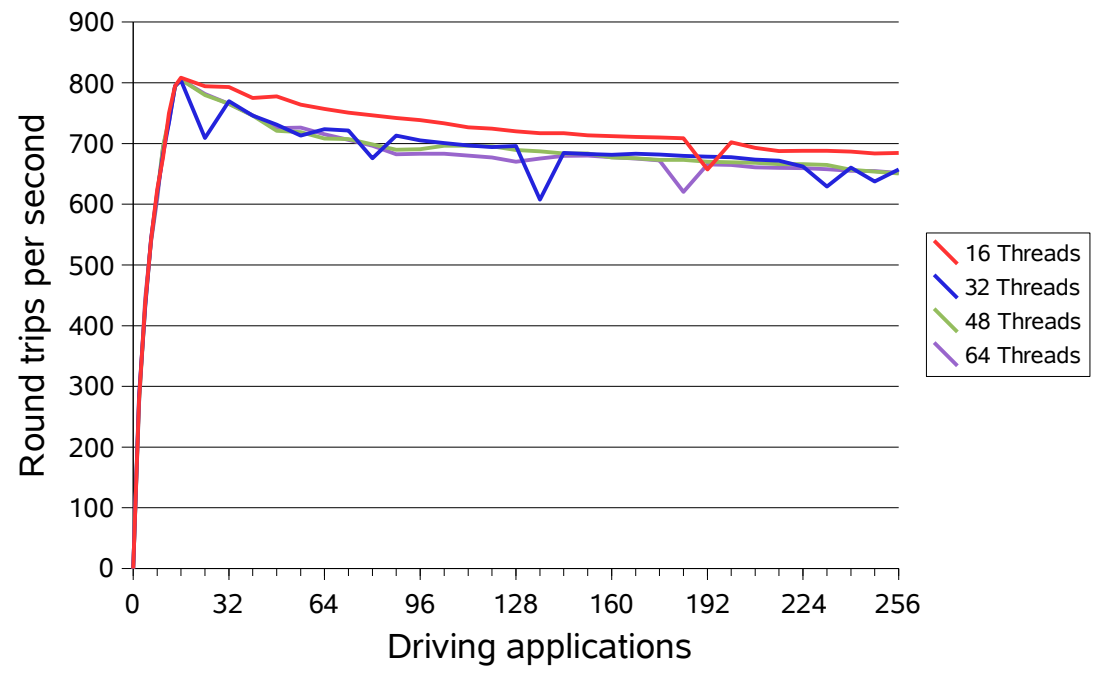


Figure 36: Persistent 20K messages, client channels between hosts, external disks

200K messages

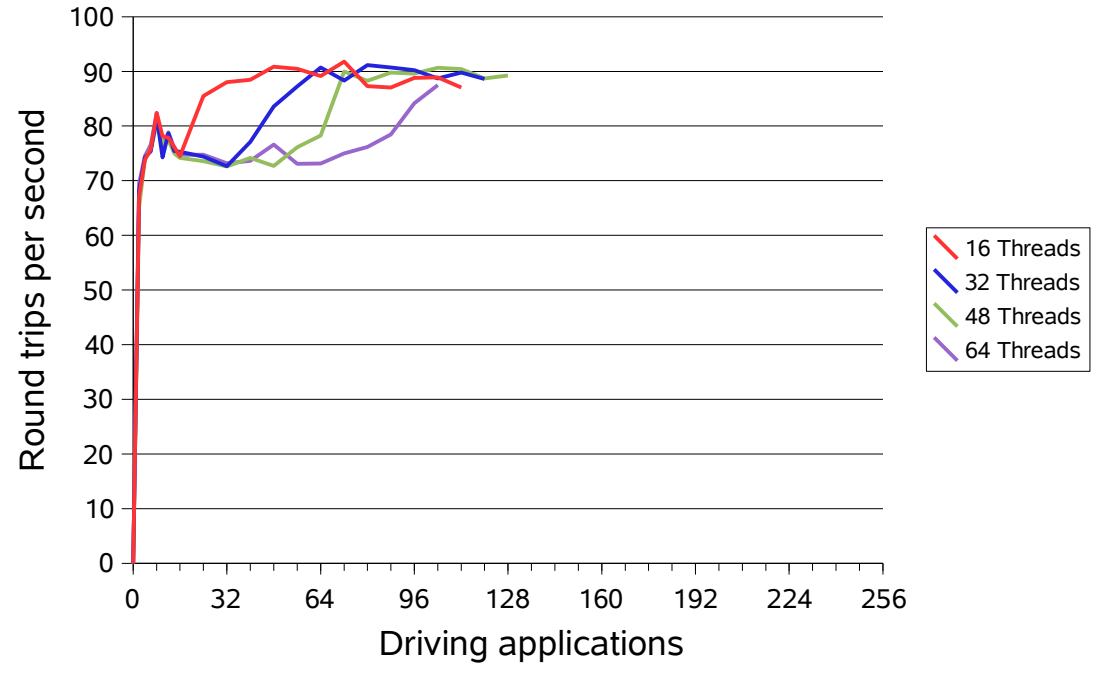


Figure 37: Persistent 200K messages, client channels between hosts, external disks

2K, 20K & 200K messages

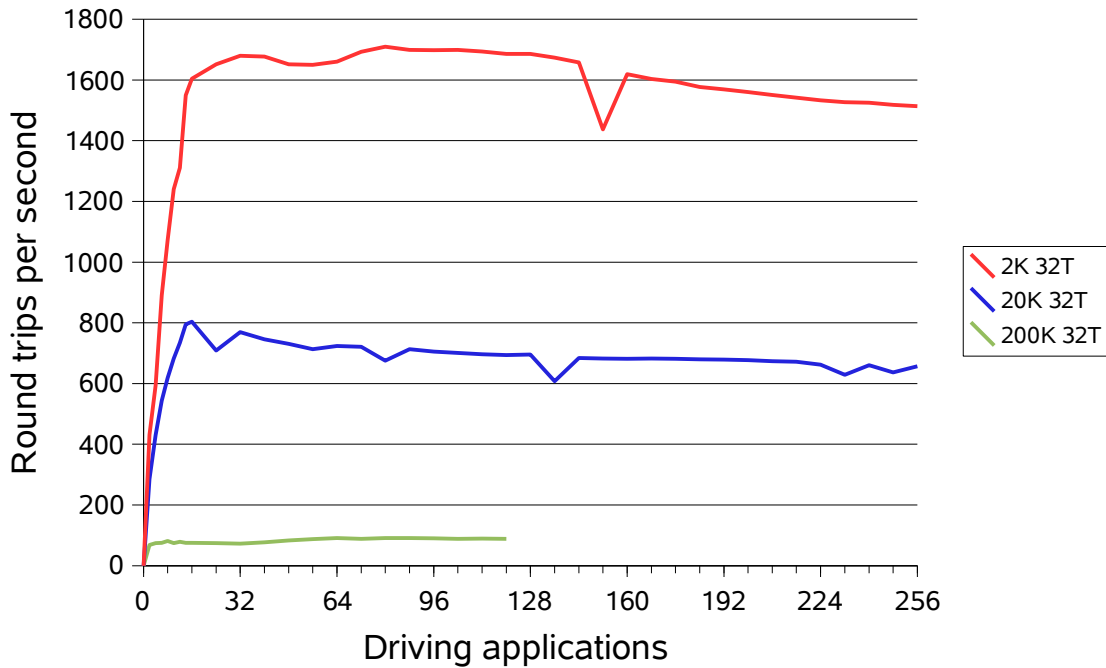


Figure 38: Comparison of 2K, 20K & 200K persistent messages, client channels between hosts, external disks

1K - 256K messages

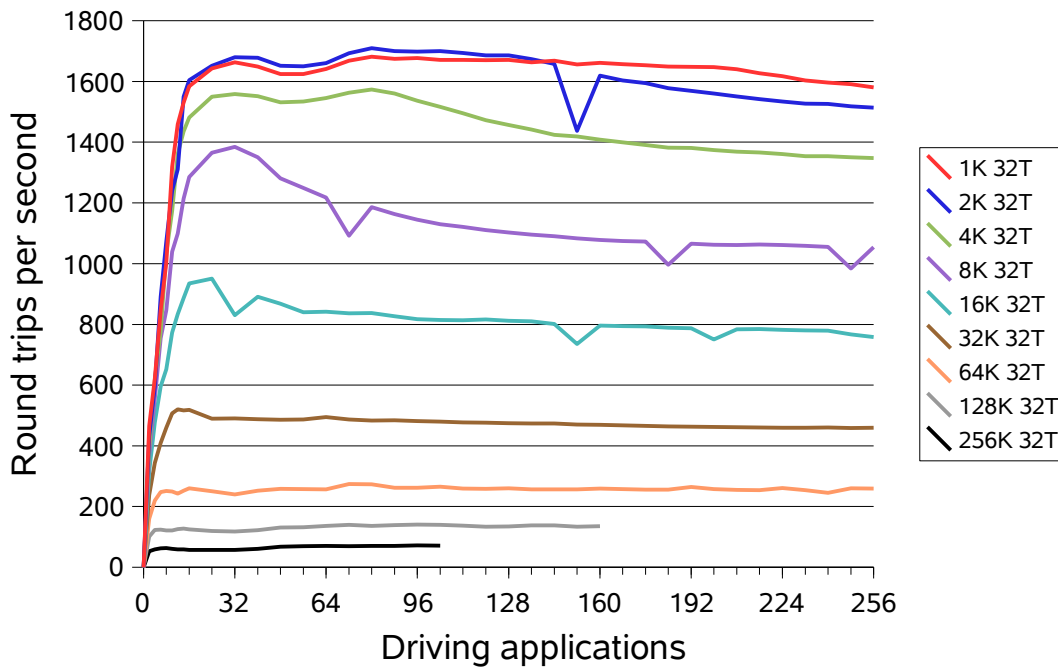


Figure 39: Comparison of persistent messages from 1K to 256K, client channels between hosts, external disks

2.3 Client channels test scenario between zones

See section 1.3 for a description of this test scenario.

2.3.1 Non persistent large messages – client channels between zones

Test name – clnp6

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the non persistent client channels between zones scenario are shown in Figure 40 (page 41) and Figure 41 (page 41).

Figure 42 (page 42) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 43 (page 42) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 18 below shows the peak non persistent message throughput observed for 2K, 20K and 200K messages in the . client channels between zones scenario

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	18	13221	0.002	93%
20K	32	24	8515	0.003	97%
200K	32	14	913	0.018	60%

Table 18: Peak non persistent 2K, 20K & 200K message performance, client channels between zones

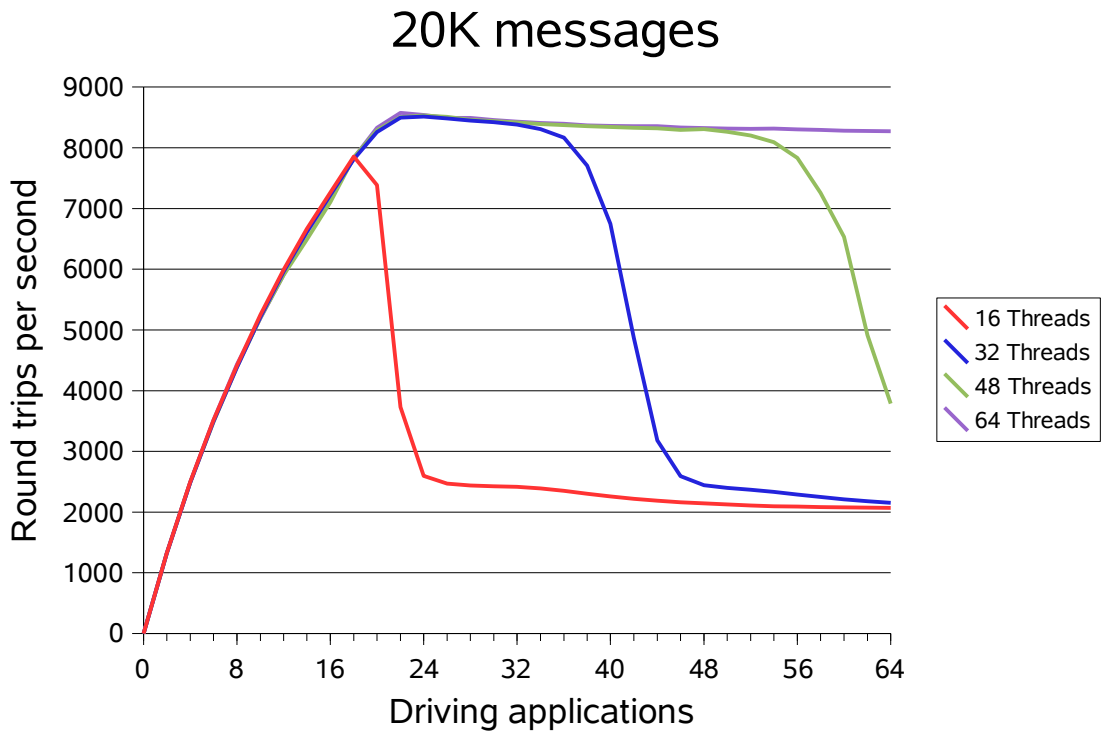


Figure 40: Non persistent 20K messages, client channels between zones

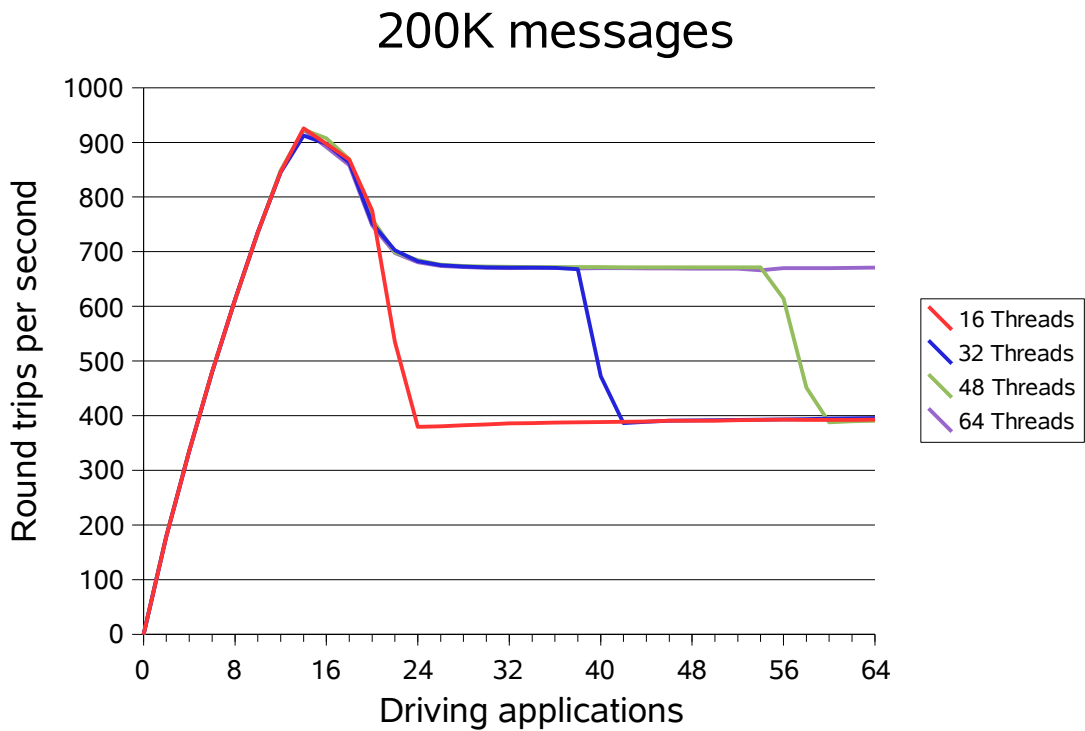


Figure 41: Non persistent 200K messages, client channels between zones

2K, 20K & 200K messages

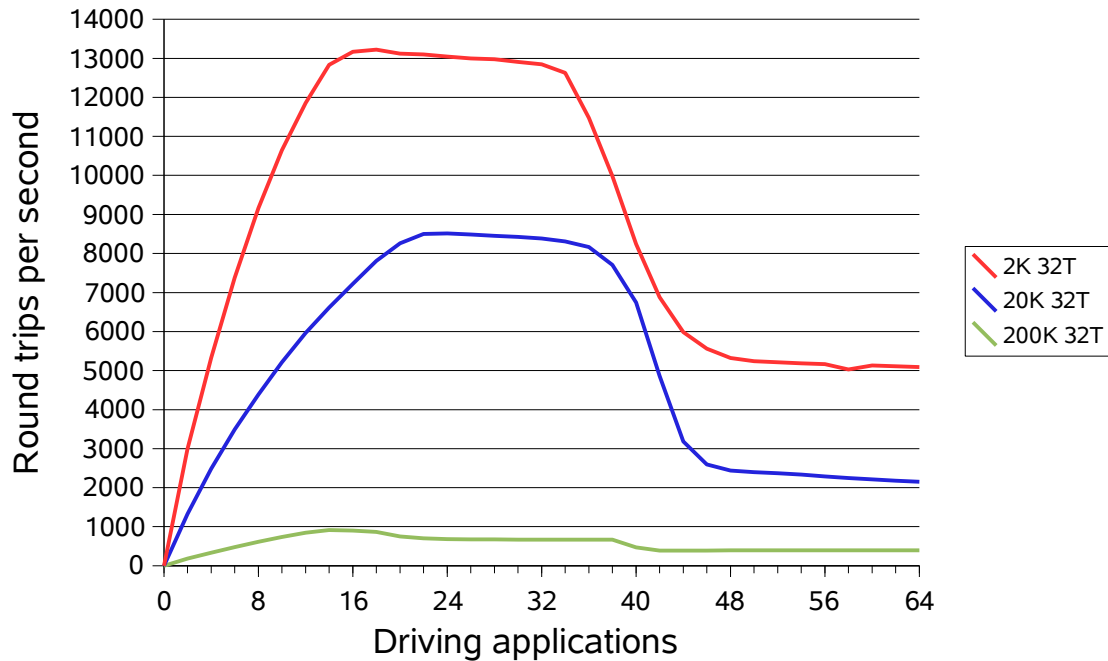


Figure 42: Comparison of 2K, 20K & 200K non persistent messages, client channels between zones

1K - 256K messages

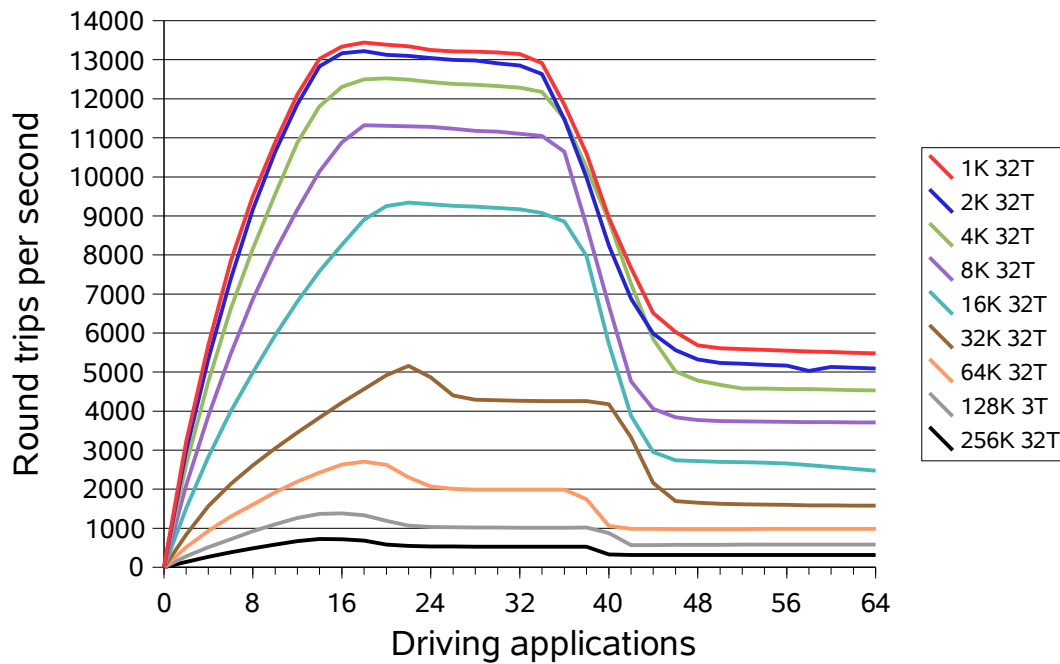


Figure 43: Comparison of non persistent messages from 1K to 256K, client channels between zones

2.3.2 Persistent large messages – client channels between zones – internal disks

This scenario was not tested.

2.3.3 Persistent large messages – client channels between zones – external disks

Test name – clpm6

Queue Manager Configuration

LogPrimaryFiles = 4, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using external disks are shown in Figure 44 (page 45) and Figure 45 (page 45).

Figure 46 (page 46) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 47 (page 46) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 19 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	16	48	1665	0.035	22%
20K	32	16	728	0.026	13%
200K	48	40	90	0.626	7%

Table 19: Peak persistent 2K, 20K & 200K message performance, client channels between zones, external disks

20K messages

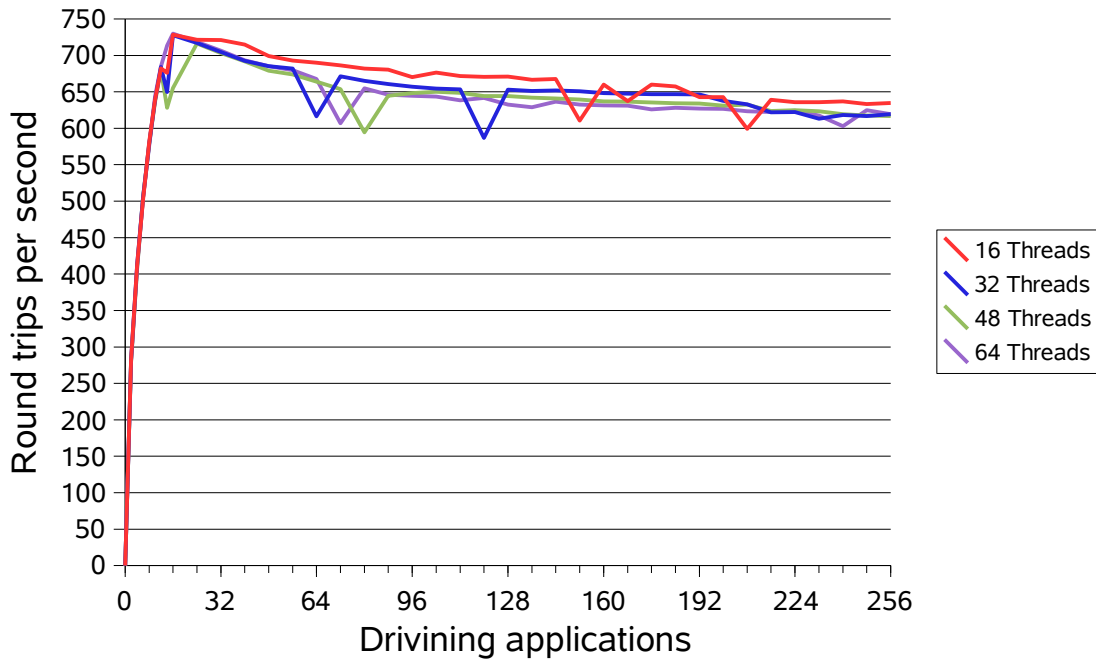


Figure 44: Persistent 20K messages, client channels between zones, external disks

200K messages

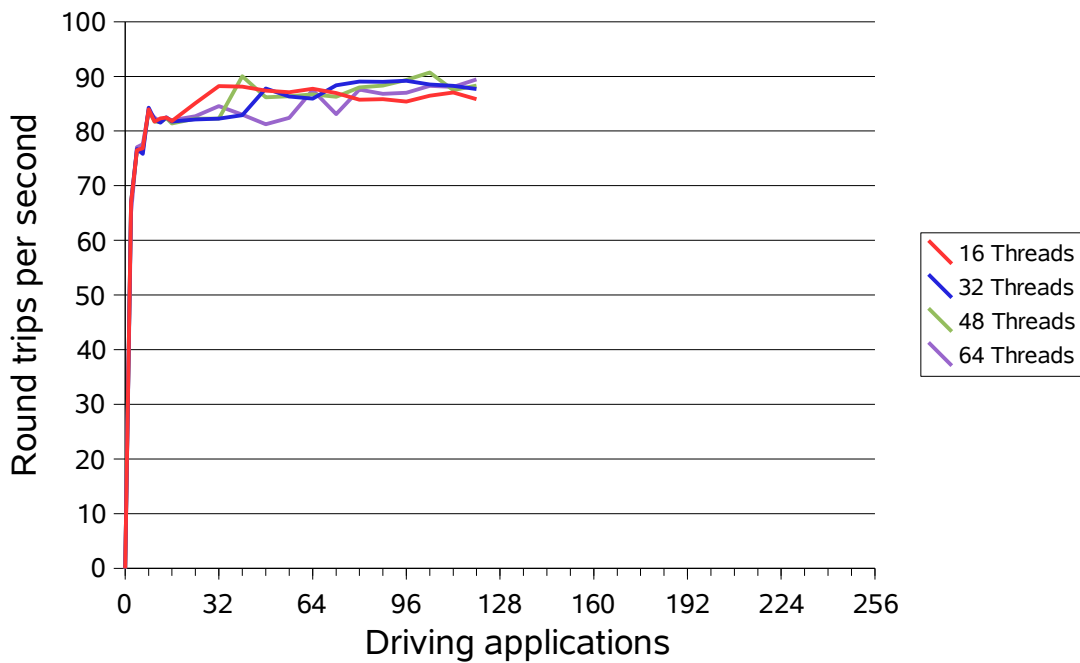


Figure 45: 200K persistent messages, client channels between zones, external disks

2K, 20K & 200K messages

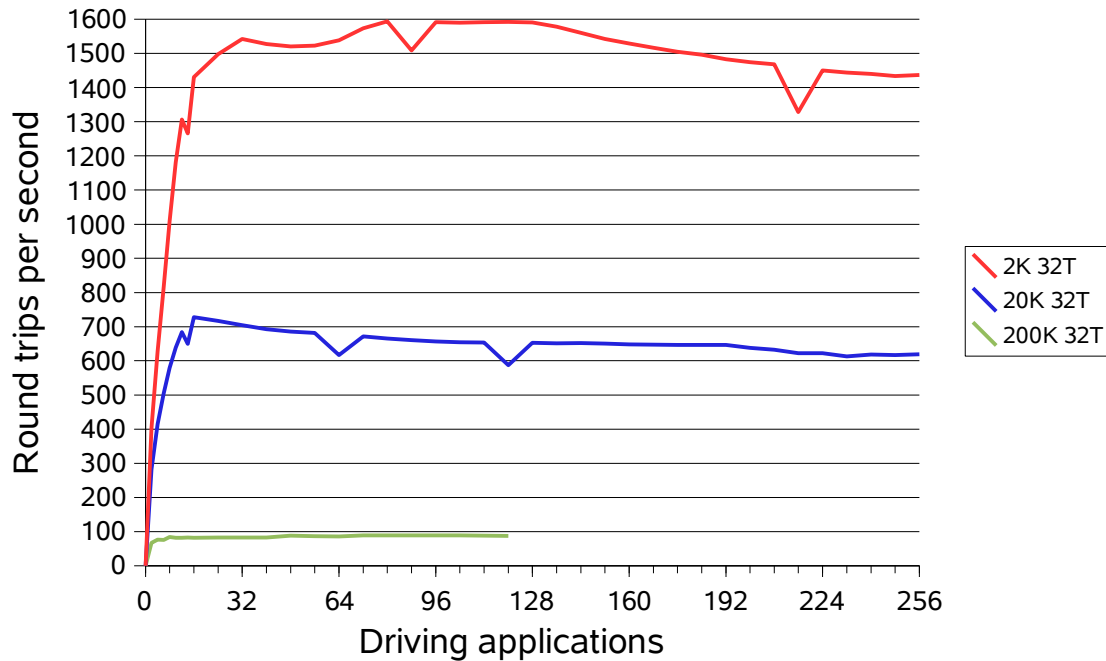


Figure 46: Comparison of 2K, 20K & 200K persistent messages, client channels between zones, external disks

1K - 256K messages

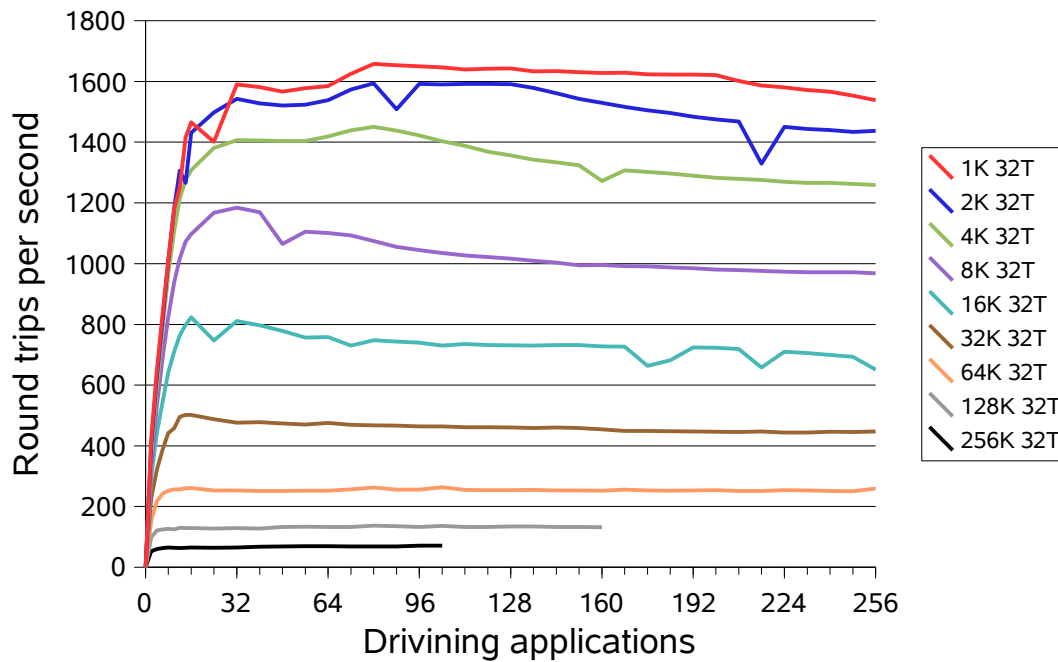


Figure 47: Comparison of persistent messages from 1K to 256K, client channels between zones, external disks

2.4 Distributed channels test scenario

See section 1.4 for a description of this test scenario.

2.4.1 Non persistent large messages – server channels

Test name – dqnp6

Queue Manager Configuration

LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using external disks are shown in Figure 48 (page 49) and Figure 49 (page 49).

Figure 50 (page 50) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 51 (page 50) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 20 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	54	13970	0.004	68%
20K	32	32	3905	0.009	31%
200K	16	40	337	0.160	17%

Table 20: Peak non persistent 2K, 20K & 200K message performance, server channels

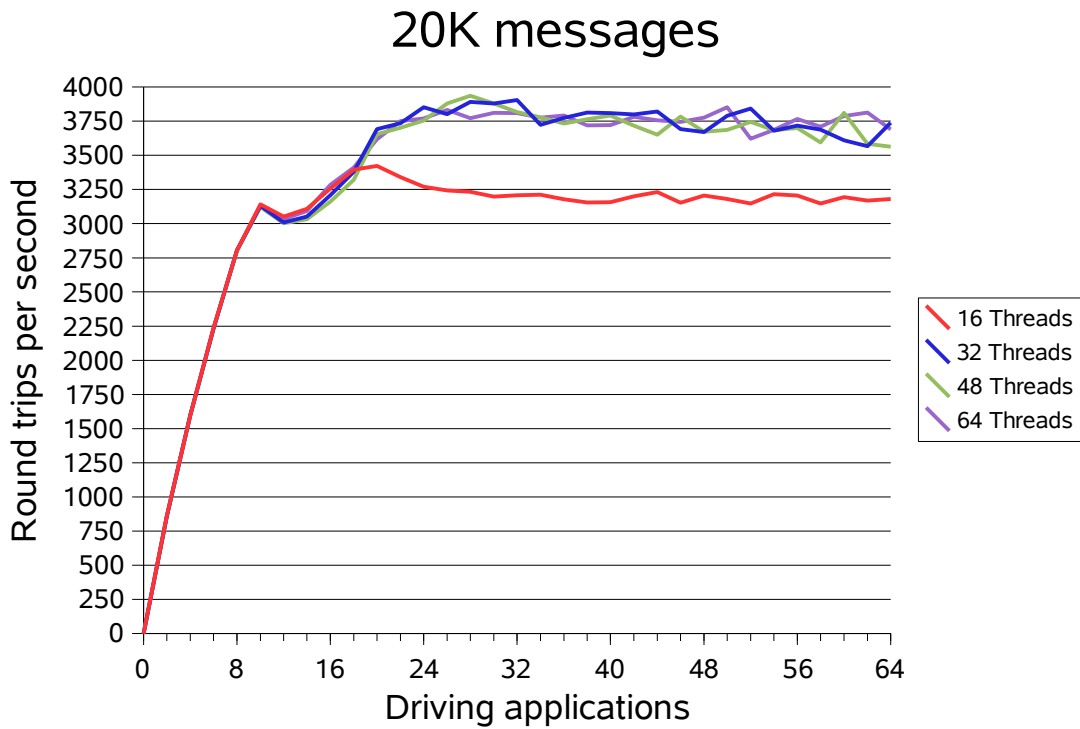


Figure 48: Non persistent 20K messages, server channels

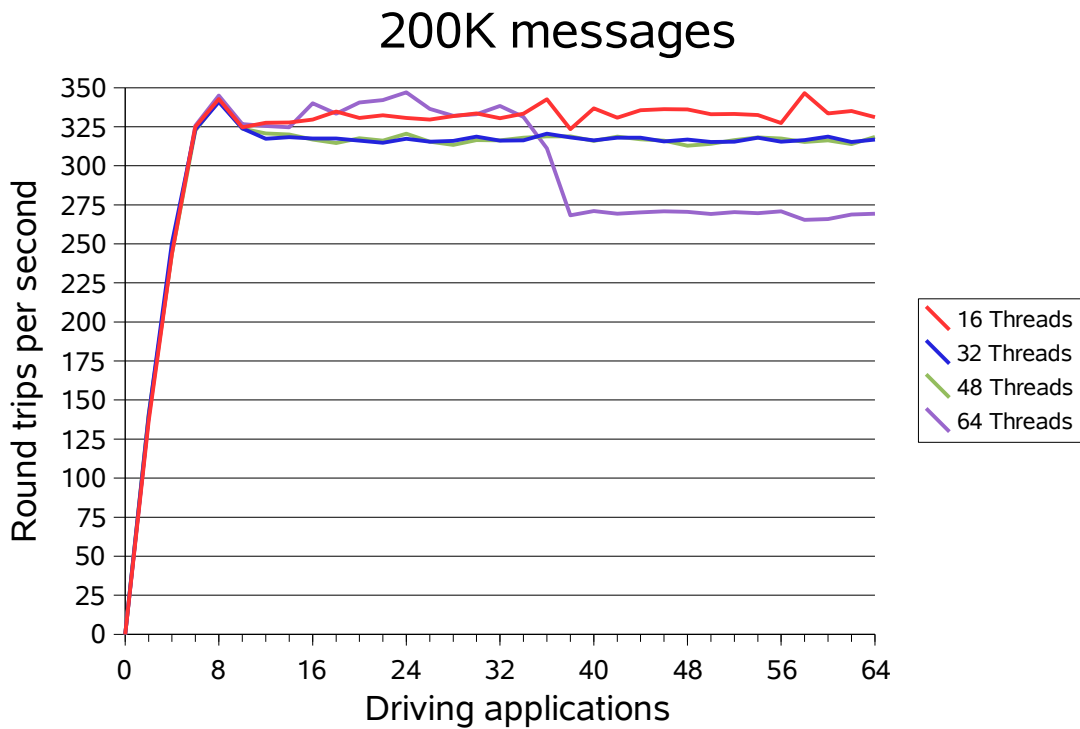


Figure 49: Non persistent 200K messages, server channels

2K, 20K & 200K messages

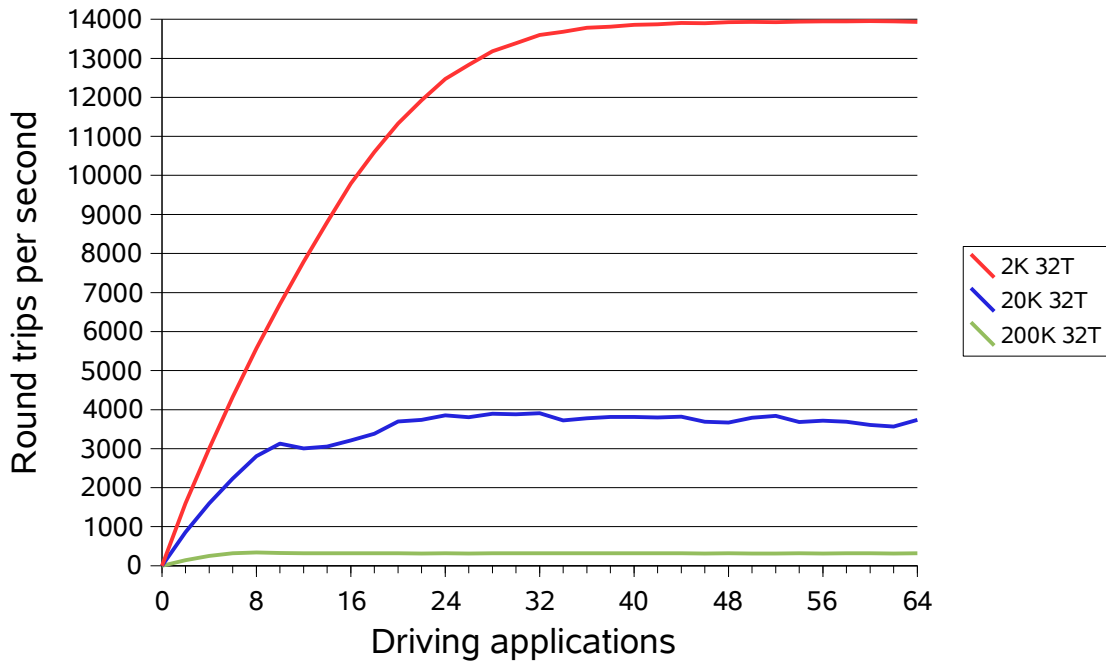


Figure 50: Comparison of non persistent 2K, 20K & 200K messages, server channels

1K - 256K messages

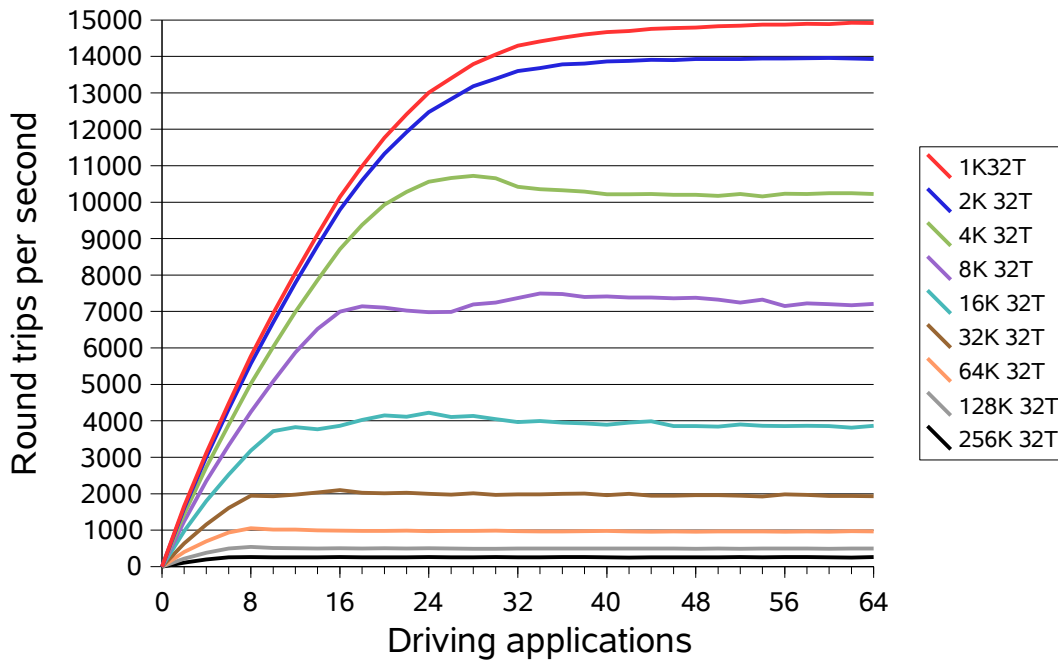


Figure 51: Comparison of non persistent messages from 1K to 256K, server channels

2.4.2 Persistent large messages – server channels – internal disks

Test name – dqpm6

Queue Manager Configuration

LogPrimaryFiles = 16, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using external disks are shown in Figure 52 (page 52) and Figure 53 (page 52).

Figure 55 (page 53) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 55 (page 53) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 21 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	32	256	1268	0.230	9%
20K	16	184	244	0.881	3%
200K	64	8	24	0.381	2%

Table 21: Peak persistent 2K, 20K and 200K message performance, server channels, internal disks

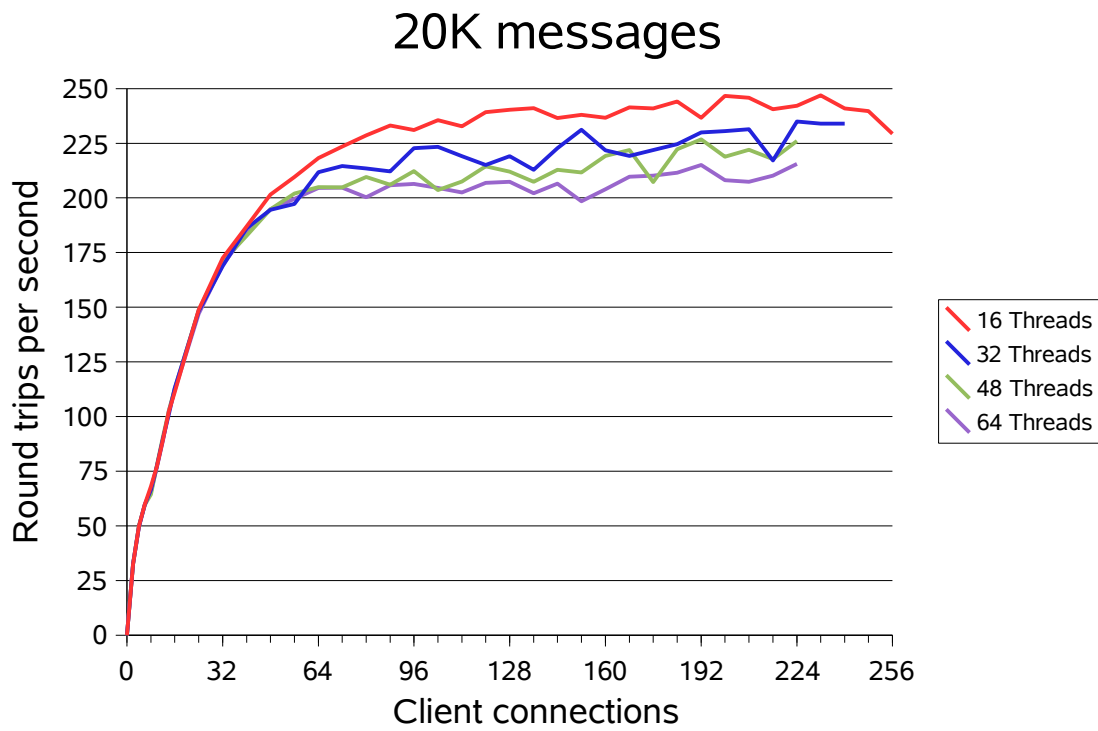


Figure 52: Persistent 20K messages, server channels, internal disks

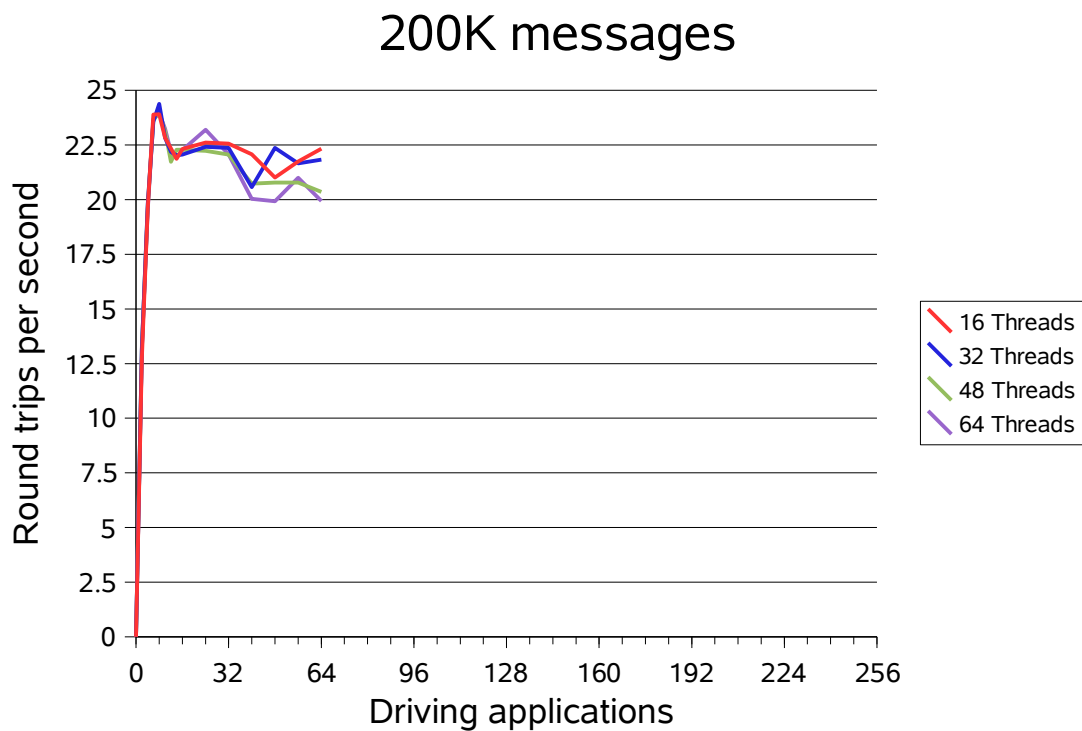


Figure 53: Persistent 200K messages, server channels, internal disks

2K, 20K & 200K messages

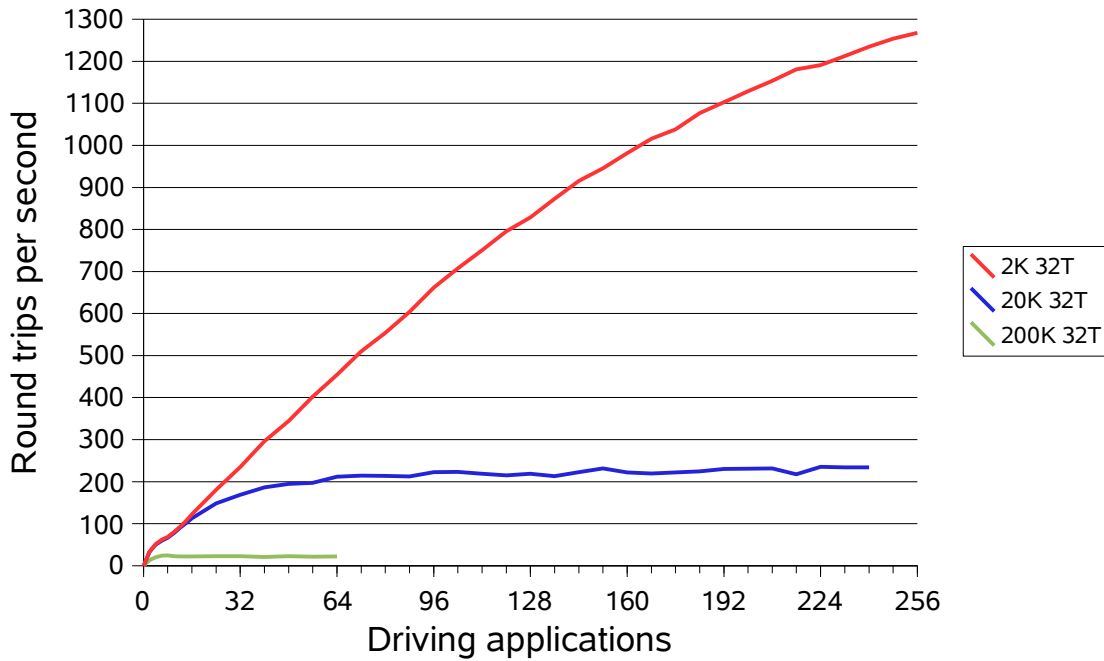


Figure 54: Comparison of 2K, 20K & 200K persistent messages, server channels, internal disks

1K - 256K messages

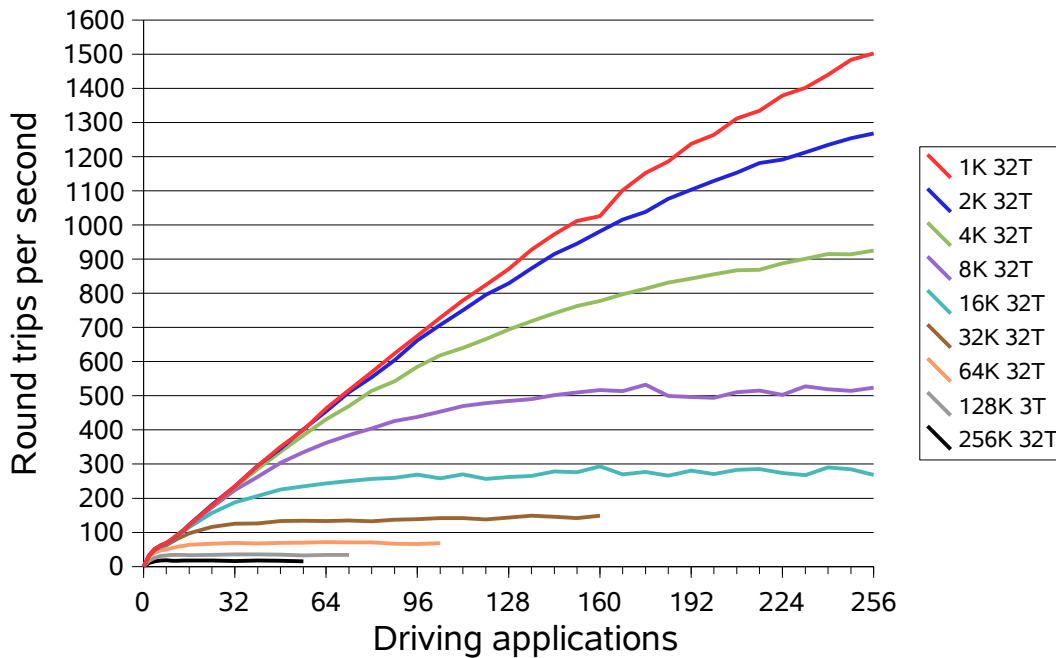


Figure 55: Comparison of persistent messages from 1K to 256K, server channels, internal disks

2.4.3 Persistent large messages – server channels – external disks

Test name – dqpm6

Queue Manager Configuration

LogPrimaryFiles = 16, LogFilePages = 16384, LogBufferPages = 512

The throughput of 20K and 200K persistent messages with an increasing number of driving applications observed with 16, 32, 48 and 64 server threads in the persistent local queue manager scenario using external disks are shown in Figure 56 (page 55) and Figure 57 (page 55).

Figure 58 (page 56) illustrates the difference in throughput of 2K, 20K and 200K messages with 32 server threads. Figure 59 (page 56) shows the effect on message throughput of doubling the message size for 1K to 256K with 32 server threads.

Table 22 below shows the peak persistent message throughput observed for 2K, 20K and 200K messages in the local queue manager scenario using external disks.

<i>Message size</i>	<i>Server threads</i>	<i>Driving applications</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU usage</i>
2K	16	224	2409	0.107	19%
20K	16	256	703	0.436	10%
200K	16	16	60	0.313	5%

Table 22: Peak persistent 2K, 20K & 200K message performance, server channels, external disks

20K messages

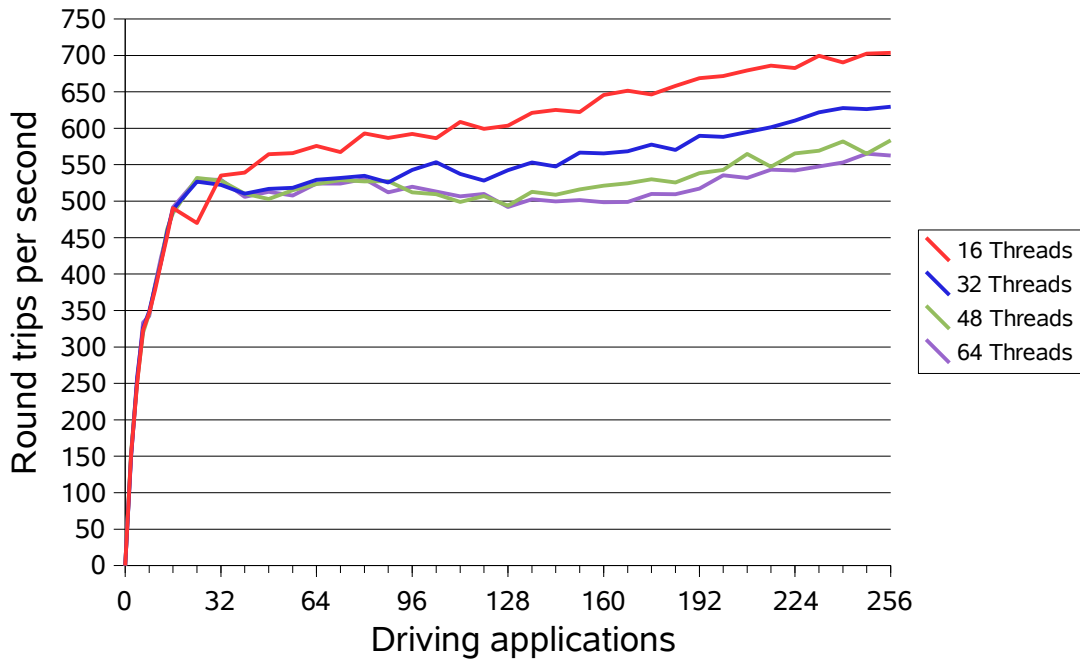


Figure 56: Persistent 20K messages, server channels, external disks

200K messages

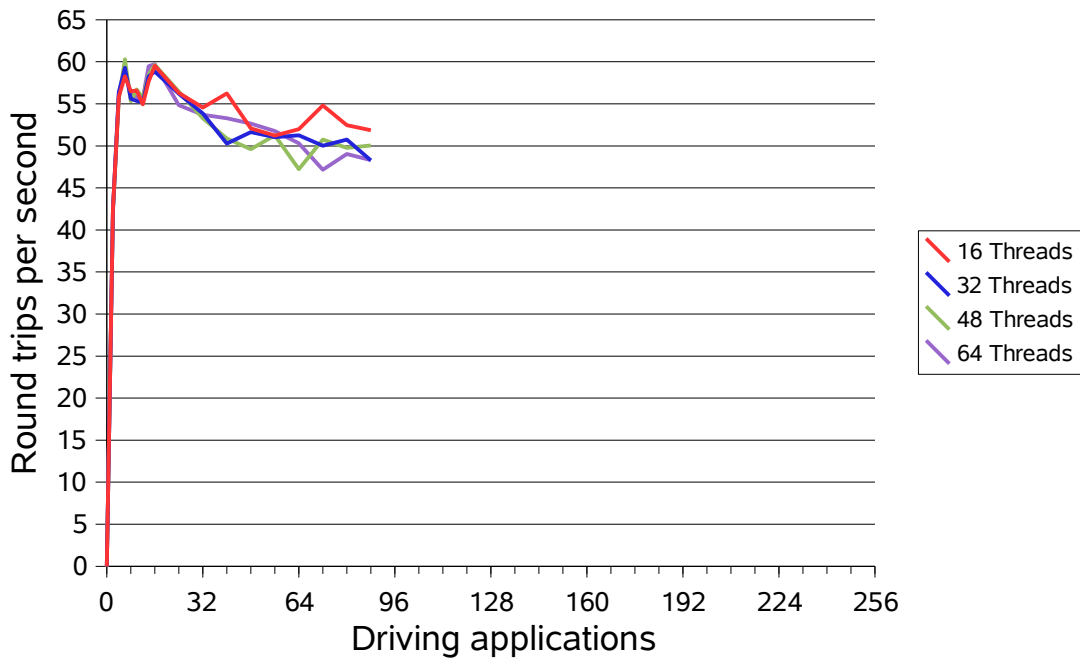


Figure 57: Persistent 200K messages, server channels, external disks

2K, 20K & 200K messages

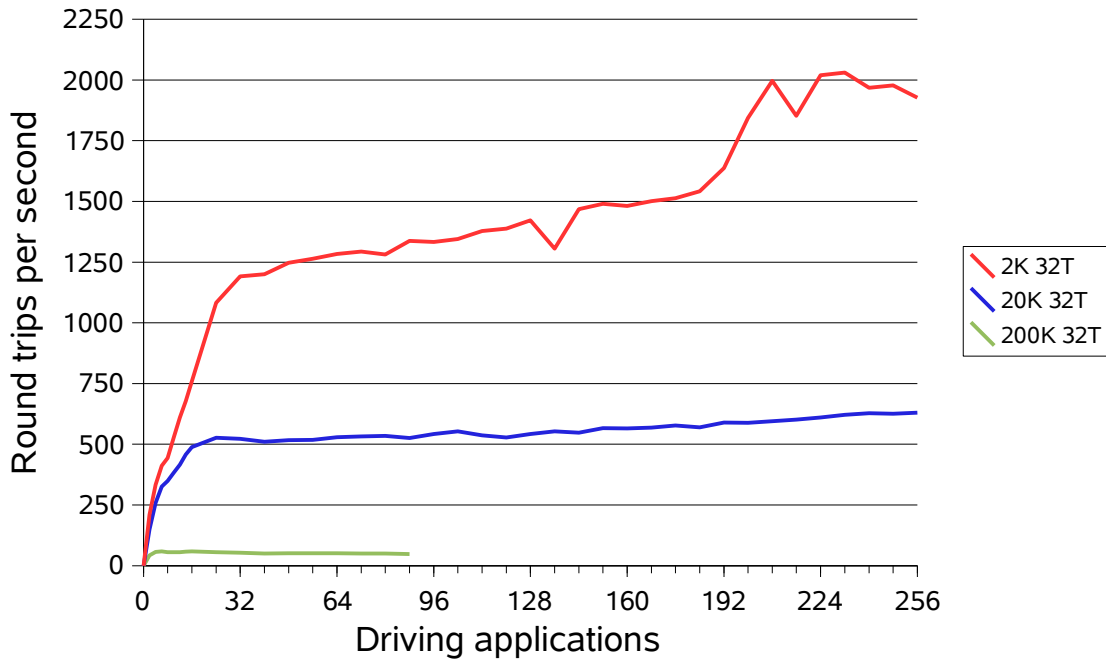


Figure 58: Comparison of 2K, 20K & 200K persistent messages, server channels, external disks

1K - 256K messages

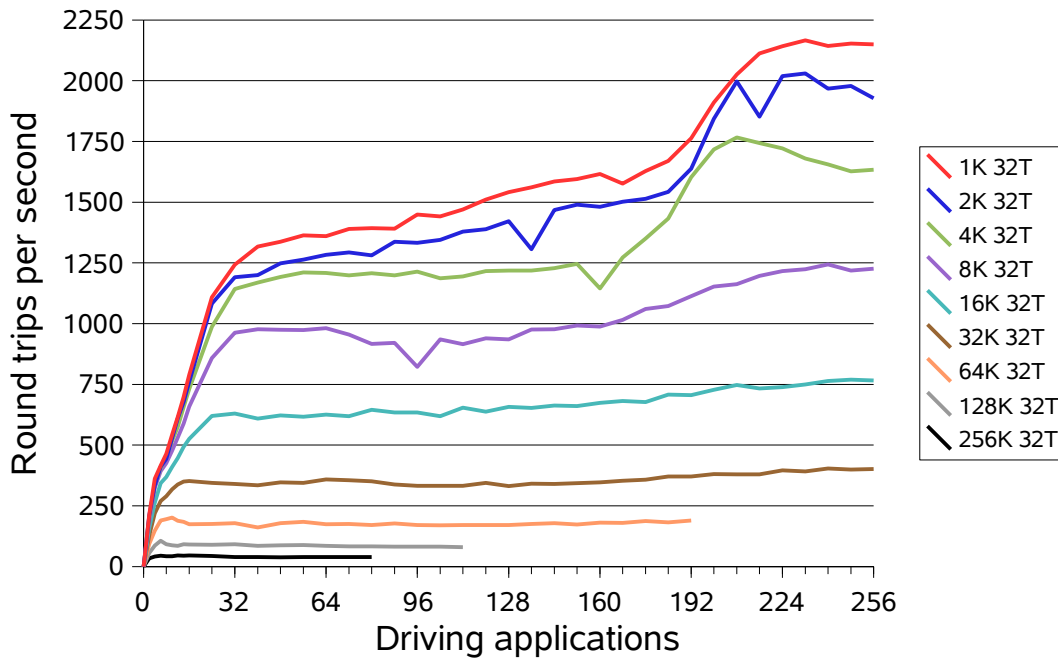



Figure 59: Comparison of persistent messages from 1K to 256K, server channels, external disks

3 Short sessions

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

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

“Why measure short sessions?”

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

This effect is greater than that of reducing the total messaging throughput of the queue manager by connecting thousands of MQI applications to the queue manager

Test name – clnp4_ss_r3600_runmqlsr

Queue manager configuration

ServerThreads = 32, LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

<i>Driving applications</i>	<i>Short sessions per sec</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU Usage</i>
1800	344	1721	0.664	30%

Table 23: Peak short session rate

Short Sessions

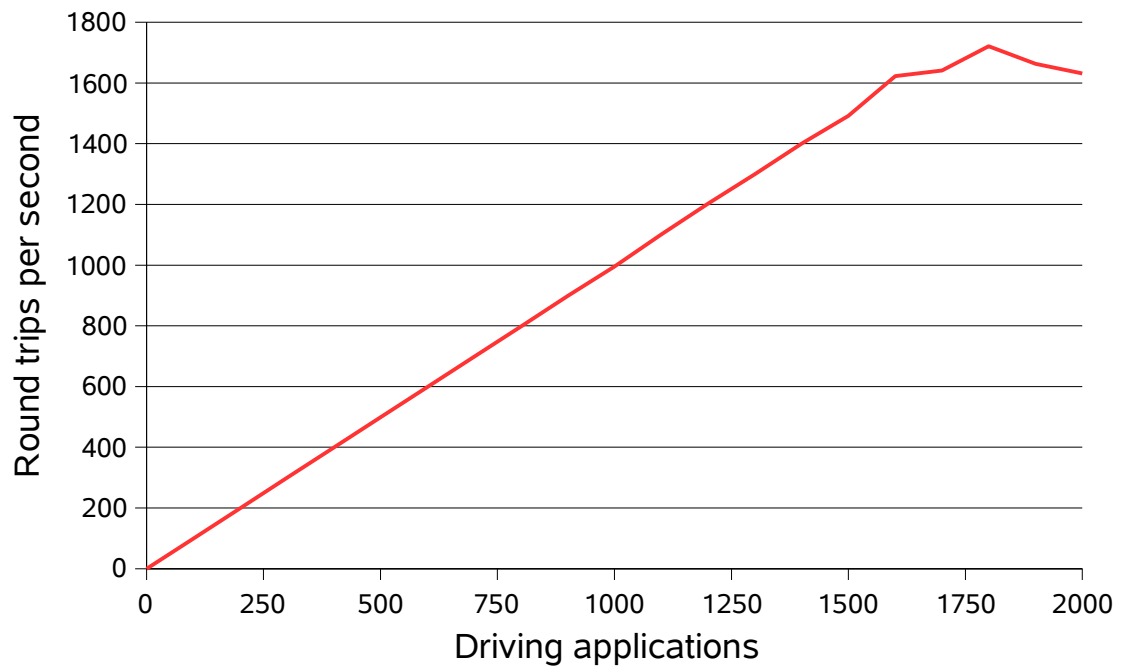


Figure 60: Short sessions

4 Performance and capacity limits

4.1 Client channels scenario – capacity measurements

The measurements in this section are intended to test the maximum number of MQI-clients that can be connected into a single queue manager with a message rate of 1 round trip per client channel per *minute*. In previous SupportPacs, the rate used in capacity limit tests was 1 round trip per *hour*. For the same number of client channels, a faster message rate gives a higher total message throughput over each channel. This information is intended to be more useful to the reader and assist them in projecting the results in this section to similar scenarios.

Test name – clnp4_c6000_runmqlsr

Queue manager configuration

*MaxChannels = 100000, ServerThreads = 32, LogPrimaryFiles = 3,
LogFilePages = 64, LogBufferPages = 17*

<i>Driving applications</i>	<i>Rate / App / Hr</i>	<i>Round trips per second</i>	<i>Response time</i>	<i>CPU Usage</i>
6000	1790	2956	0.009	19%

Table 24: Client channel capacity measurement

4.2 Distributed queuing scenario – capacity measurements

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

Test name – dqnp1_q1000_runmqlsr

Queue manager configuration

ServerThreads = 24, LogPrimaryFiles = 3, LogFilePages = 64, LogBufferPages = 17

<i>Driving apps.</i>	<i>Channel pairs</i>	<i>Rate / App / Hr</i>	<i>Round trips / sec</i>	<i>Response time</i>	<i>CPU Usage</i>
1000	1000	35980	2933	0.36	17%

Table 25: Distributed queuing capacity measurements

Test name – dqnp1_qmax_runmq/sr

Queue manager configuration

ServerThreads = 32, LogPrimaryFiles = 12, LogSecondaryFiles = 3
 LogFilePages = 16384, LogBufferPages = 512

Driving apps.	Channel pairs	Rate / App / Hr	Round trips / sec	Response time	%CPU
10000	10000	60	186	0.002	2%

Table 26: Distributed queuing capacity measurements

Driving applications	Swap GB	Shared Memory GB	Free GB
10000	6.4 (478K/App)	1.76 (90KB/App)	8.5

Table 27: Distrubuted queuing capacity, swap and shared memory requirements

The figure in brackets are the average amount of swap and shared memory per driving application – in this test scenario this relates to the cost of a server channel pair on the server machine.

5 Tuning recommendations

5.1 Tuning the queue manager

This section highlights the tuning activities that are known to give performance benefits for WebSphere MQ V6.0. 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 result 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 those changes that are essential. 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 a documented external interface and may change or be removed in future releases.

5.1.1 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.

Nonpersistent queue buffer

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

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

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

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

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

- improve the throughput of persistent messages (permitting a possible maximum 2MB of log records to be written from the log buffer to the log disk in a single write),
- reduce the frequency of log switching (permitting a greater amount of log data to be written into one extent),
- and 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* parameter takes effect at the next queue manager restart. The number of pages can be changed for all subsequent queue managers by changing the *LogBufferPages* parameter in the product default *Log* stanza.

It is unlikely that poor persistent message throughput will be attributed to the 2MB limit of the queue manager log buffer. 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.

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

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

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

5.2 Application design and configuration

5.2.1 *Standard* or fastpath?

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

5.2.2 Parallelism, batching, and triggering

An application should be designed wherever possible to have the capability to run *multiple instances* or *multiple threads* of execution. Although the capacity of a multi-processor (SMP) system can be fully utilised with a small number of applications using nonpersistent messages, more applications are required if the workload is mainly using persistent messages. Processing messages inside syncpoint can help reduce the amount of time the queue managers takes to write a batch of persistent messages to the log disk. The performance profile of a workload will also be subject to variability through cycles of low and high message volumes, therefore a degree of experimentation will be required to determine an optimum configuration.

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

The reader should note that as more applications are processing messages on a single queue there is an increasing likelihood that queue avoidance will not be maintainable. The reasons for this have a cumulative and exponential effect, for example, when nonpersistent messages are being placed on a queue quicker than they can be removed. The first effect is that messages begin to fill the nonpersistent queue buffer—and MQGETers need to retrieve messages from the buffer rather than being received directly from an MQPUTer. A secondary effect is that as messages are spilled from the buffer to the queue disk, the MQGETers must wait for the queue manager to retrieve the message from the queue disk rather than being retrieved from the queue buffer. While these problems can be addressed by configuring for more MQGETers (i.e. processing threads in the server application), or using a larger nonpersistent queue buffer, it may not be possible to avoid a performance degradation.

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

queue manager must wait for each message to be written to the log before returning control to the application.

A typical triggered application follows the performance profile of a short session (refer to 'Short sessions' on page 57). The 'runmqsr' has a much smaller overhead of connecting to and disconnecting from the queue manager because it does not have to create a new process. Furthermore, in Version 5.3 the maximum number of connections into a single 'runmqsr' listener has been significantly increased making it the preferred method of running short sessions over client channels. Nonetheless, the implementation of triggering is still worth consideration with regard to programming a disconnect interval as an input parameter to the application. This can provide the flexibility to make tuning adjustments in a production environment, if for instance, it is more efficient to remain connected to manager between periods of message processing, or disconnect to free queue manager and Operating System resources.

5.3 Tuning the Solaris 10 Operatings System

In Solaris 10 the System V IPC parameters are dynamic and it no longer necessary to configure them. All the test in this report were done without any specific tuning of the Solaris kernel.

6 Measurement environment

6.1 Workload description

6.1.1 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) and MQDISC,
- MQOPEN and MQCLOSE,
- MQPUT and 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.

6.1.2 Test scenarios workload

The driving application programs

The test scenario workload simulates many driving applications running on a several driving machines. This is not typical of a customer environment and is only used to facilitate test coordination. Driving applications were multi-threaded with each thread performing a sequence of MQI calls. The number of threads in each application was adjusted according to whether the test was measuring a local queue manager, a client channel, or distributed queuing scenario. This was done to reduce storage overheads on the driving system. Each driving application thread performed the sequence of actions as outlined in the test scenario illustrations in the 'Performance headlines' starting on page 1

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

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

Non persistent 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.

The server application program

The server application is written as a multi-threaded program configured to use 16, 32, 48 or 64 threads. Each server thread performed the sequence of actions as outlined in the test scenario illustrations in the 'Performance Headlines' starting on page 1

Non persistent messaging is done outside of syncopate control. Persistent messaging is done inside of syncopate control. The average message throughput expressed as a number of round trips per second was calculated and reported by the server program.

6.2 Hardware

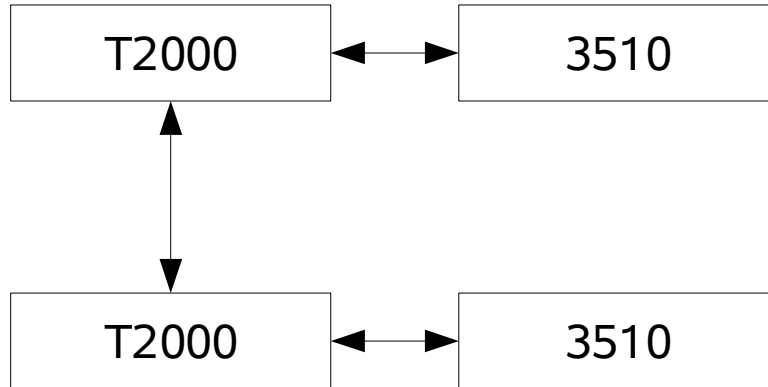


Figure 61: Hardware configuration

The test environment consists of two 1Ghz Sun Fire T2000 systems connected via a private gigabit Ethernet network. Each system has 16GB of memory. Each system has a 3510 storage array with 12 disks and a single raid controller (simulated by offlining the secondary controller). The 3510 RAID controller has 1G cache and batteries too maintain the cache for upto 72 hours in the event of a power failure. Each 3510 was configured to export two RAID 0 LUN's each of which contained 6 disks. All configurable options for the 3510 were left at their default values.

Persistent message throughput was measured using the T2000's internal SAS (Serial Attach SCSI) disks and with an external 3510 fibre channel RAID array. When using the internal disks, one drive was dedicated to the queue manager (mounted on /var/mqm) and a second drive was dedicated to the logs (mounted on /var/mqm/log). When using the external disks, one LUN was used for the queue manager and the second was used for the logs.

6.3 Software

Software	Version
Solaris OS	Solaris 10
Solaris Kernel Patch	118833-03
Compiler	Sun Studio 11
MQ Series	V6.00.0

Table 28: Software

7 Glossary

Test name	<p>The name of the test</p> <p>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:</p> <p>local => local queue manager test scenario</p> <p>cl => client channel test scenario</p> <p>dq => distributed queuing test scenario</p> <p>np => nonpersistent messages</p> <p>pm => persistent messages</p> <p>r3600 => 1 round trip per driving application per second</p> <p>runmqlsr => channels using the 'runmqlsr' listener (client channel test scenario, in addition to 'runmqchi' for distributed queuing test scenarios)</p> <p>c6000 => 6,000 client driving applications (i.e. 6,000 MQI-client connections)</p> <p>q1000 => 1,000 server channel pairs</p> <p>max => maximum number of channels (or channel pairs)</p> <p>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)</p> <p>t10 => indicates 10 threads per driving application</p>
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
Resp time (s)	The average response time each round trip, as measured and averaged by all the driving applications
Swap	The total amount of swap space that is either allocated or reserved
shm	The amount of allocated System V IPC shared memory in MB
segs	The number of System V IPC shared memory segments
sems	The number of System V IPC semaphores

END OF DOCUMENT