WebSphere®MQ Publish Subscribe V7.5 Performance Evaluations

Version 1.0

October 2012

Rachel Norris

WebSphere MQ Performance
IBM Hursley

IBM United Kingdom Limited

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, October 2012.

This edition applies to *WebSphere MQ V7.5* (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 Publish Subscribe V7.5*. 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 V7.5

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

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

This report presents the results of performance evaluations of WebSphere MQ V7.5and is intended to assist with capacity planning.
The results in this report can also be assumed to be valid for WebSphere MQ V7.1 since there are no significant performance differences between the two releases.

Target audience

This SupportPac is designed for people who:

- Will be designing and implementing Publish Subscribe solutions using WebSphere MQ.
- Want to understand the performance of WebSphere MQ Publish Subscribe.

The reader should have a general awareness of the Publish Subscribe API, AIX, Linux, and/or Windows operating systems and of WebSphere MQ in order to make best use of this SupportPac.

The contents of this SupportPac

This SupportPac includes:

- Charts and tables describing the performance of WebSphere MQ V7.5 Publish Subscribe

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.

Contents

Charts

1 Overview

Most of the test scenarios in this report are measured in four configurations :-

1) Local applications, non-persistent messages, non-transacted, non-durable subscriptions
2) Local applications, persistent messages, transacted, durable subscriptions.
3) Client applications, non-persistent messages, non-transacted, non-durable subscriptions
4) Client applications, persistent messages, transacted, durable subscriptions

On three operating systems:-

1) AIX
2) Linux64
3) Windows

In the Local test configuration the test applications are located on the same machine as the WebSphere MQ Queue Manager and use Standard Bindings.

In the Client test configuration the test applications are located on separate machines from the WebSphere MQ Queue Manager. Multiple client machines are used to ensure that the WebSphere MQ server machine is driven to its maximum capacity. A 10Gb LAN is used to connect the client machines to the server, except for the Windows tests where a 1Gb LAN was used because the server has a small number of cores and is not network limited.

Persistent message tests are transactional as this provides the highest level of reliability. The speed of disks which the messages and logs are written to is a key factor in the performance of persistent tests. SAN disks with write cache were used for all tests in this report to ensure low latency I/O.

Non-persistent tests create non-durable subscriptions and Persistent tests create durable subscriptions.

All subscriptions use managed queues which means that a subscription queue is automatically created by WebSphere MQ when a subscription is created.

The tests in the report use a 2KB message unless otherwise specified.

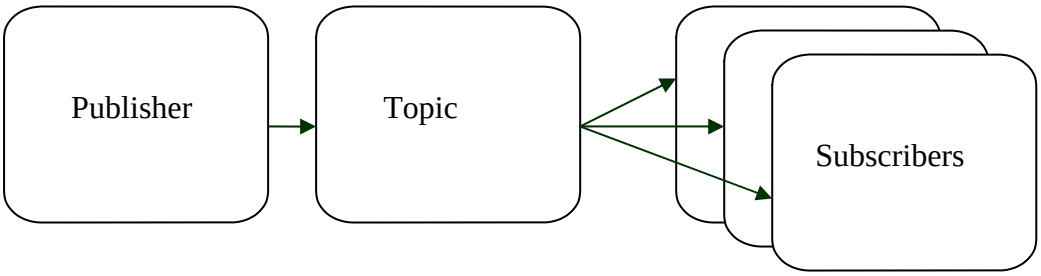
The tests use a C test client written to use the MQI. The IBM Performance Harness for JMS was used as the test client for the JMS comparison section.

The message rates published in this report are dependant on the hardware used to run the tests. Different hardware was used for each operating systems and hence direct comparisons between operating systems cannot be made. For details of the hardware used see section Machine and Test Configurations

2 Test scenarios with AIX results

2.1 Affect of number of subscriptions on publication rate

In this test scenario there is a single Publisher publishing messages on a single Topic. The number of Subscribers to the topic is gradually increased during the test run. Each subscriber receives a copy of each message published by the single publisher.



The graphs below show how the Publication rate and total message rate are affected by increasing the number of subscriptions. The total message rate is the sum of the messages published and the messages consumed by the subscribers.

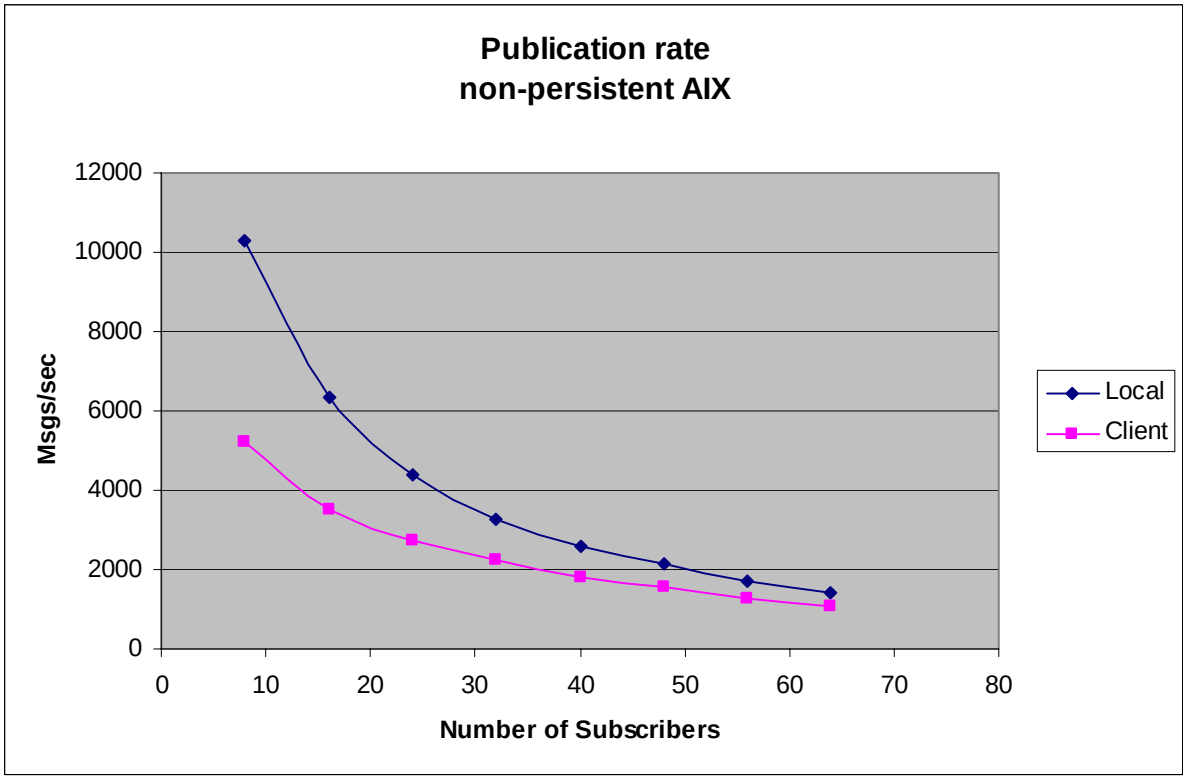


Chart 1 - Publication rate non-persistent AIX

Chart 16 - Publication rate non-persistent Linux64
Chart 28 - Publication rate non-persistent Windows

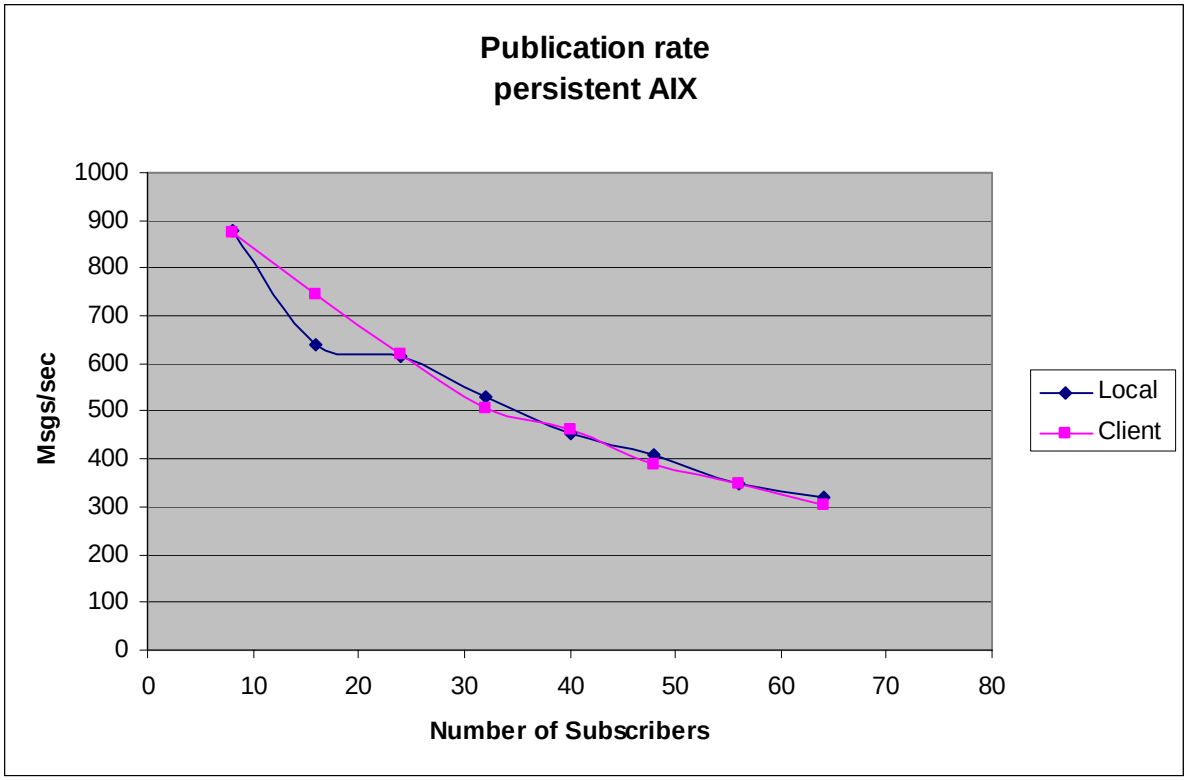


Chart 2 – Publication rate persistent AIX

Chart 17 - Publication rate persistent Linux64
Chart 29 - Publication rate persistent Windows

The results show that as the number of subscriptions increases the rate at which the Publisher can publish messages falls. This is because each publication is delivered to all subscriptions on the topic before the next message can be published by that publisher.

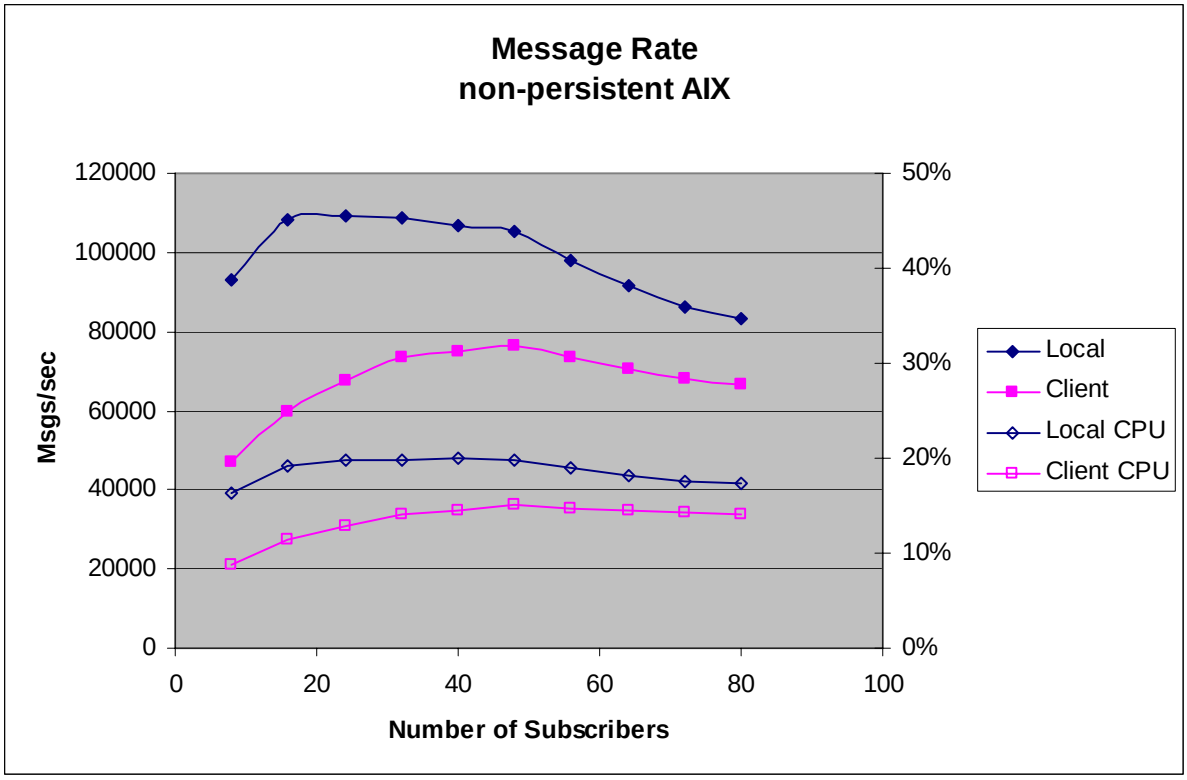


Chart 3 – Message rate non-persistent AIX

Chart 18 - Message rate non-persistent Linux64
Chart 30 - Message rate non-persistent Windows

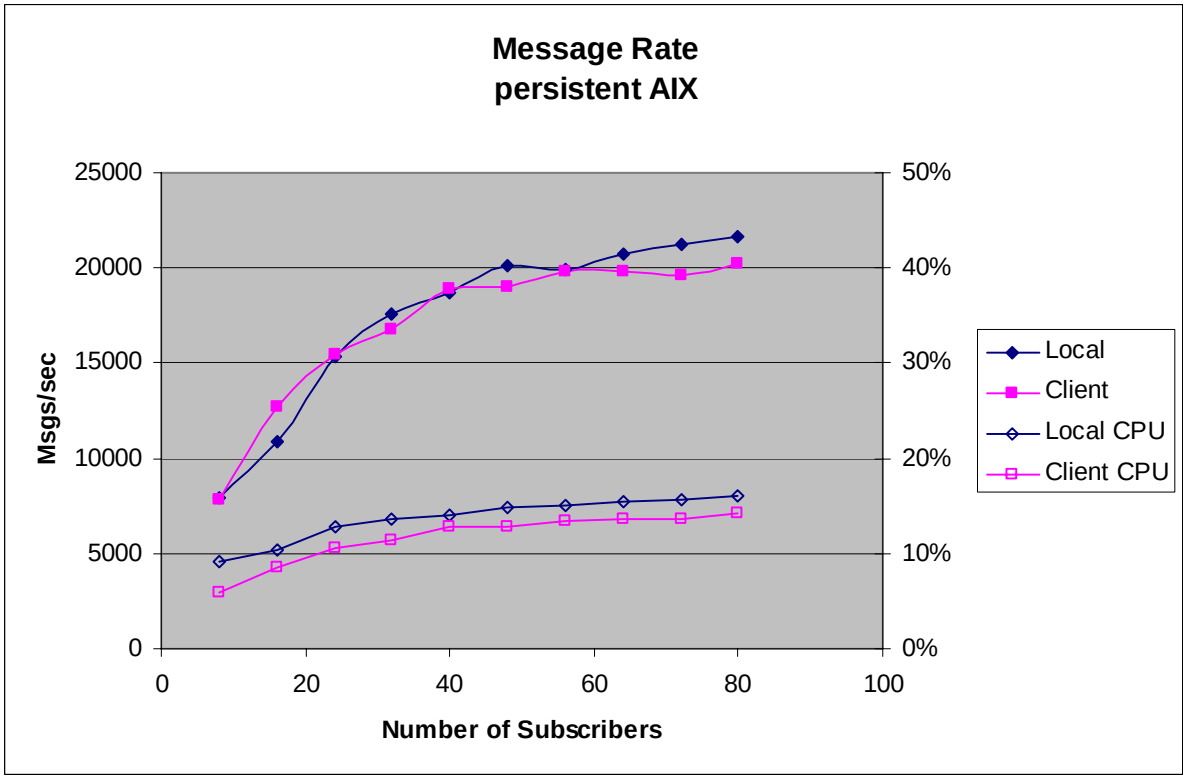


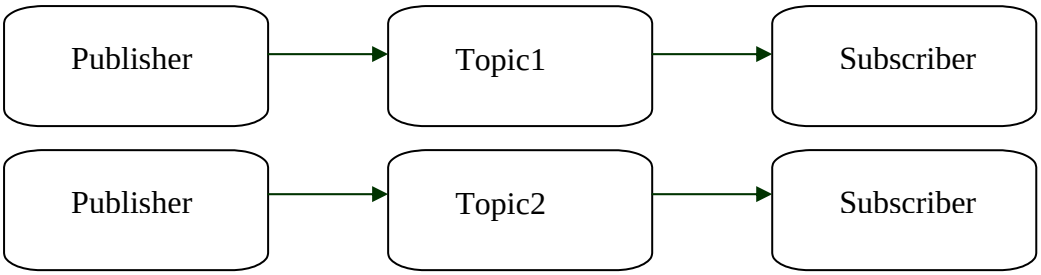
Chart 4 - Message rate persistent AIX

Chart 31 - Message rate persistent Windows

The total message rate increases until the limit is reached. Note that this is not the maximum message rate which the Queue Manager can sustain because the rate and CPU usage are limited by having a single Publisher.

2.2 Maximum Publish/Subscribe message rate

In this test scenario there is one Publisher and one Subscriber on each topic. The number of publishers, subscribers and topics is gradually increased during the test run. The publishers publish messages at a fixed rate. For the non-persistent tests the rate is 1600 msgs/sec per publisher and for persistent tests the rate is 400 msgs/sec per publisher. The aim of this scenario is to drive the Queue Manager to the maximum message rate. The results are intended for capacity planning use.



The graphs below show how the total message rate increases until the maximum rate possible on the Queue Manager is reached.

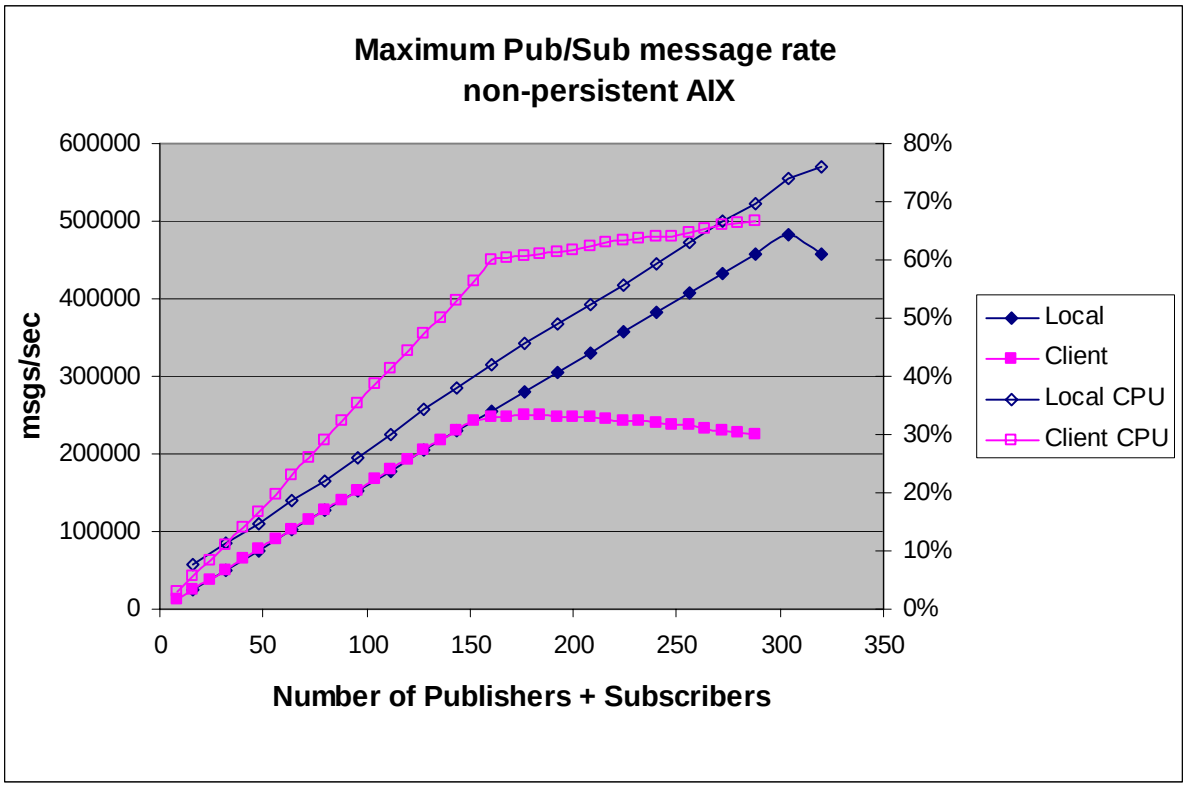


Chart 5 – Maximum Pub/Sub message rate non-persistent AIX

Chart 20 - Maximum Pub/Sub message rate non-persistent Linux64
Chart 32 - Maximum Pub/Sub message rate non-persistent Windows

The maximum throughput in the Client scenario is limited by the network. This is not because the network bandwidth has been exceeded, but is caused by WebSphere MQ sending a confirmation message to the publisher for every message published. This means that the network switches between inbound and outbound traffic for every message, which is inefficient.

It is possible to change this behaviour by using asynchronous put (WebSphere MQ v7.5 Information Center - asynchronous put). When an application publishes a message using asynchronous put the queue manager does not send a success or failure message but instead the application can check for errors periodically. The graph below shows the Maximum Pub/Sub client message rate achievable using asynchrous put.

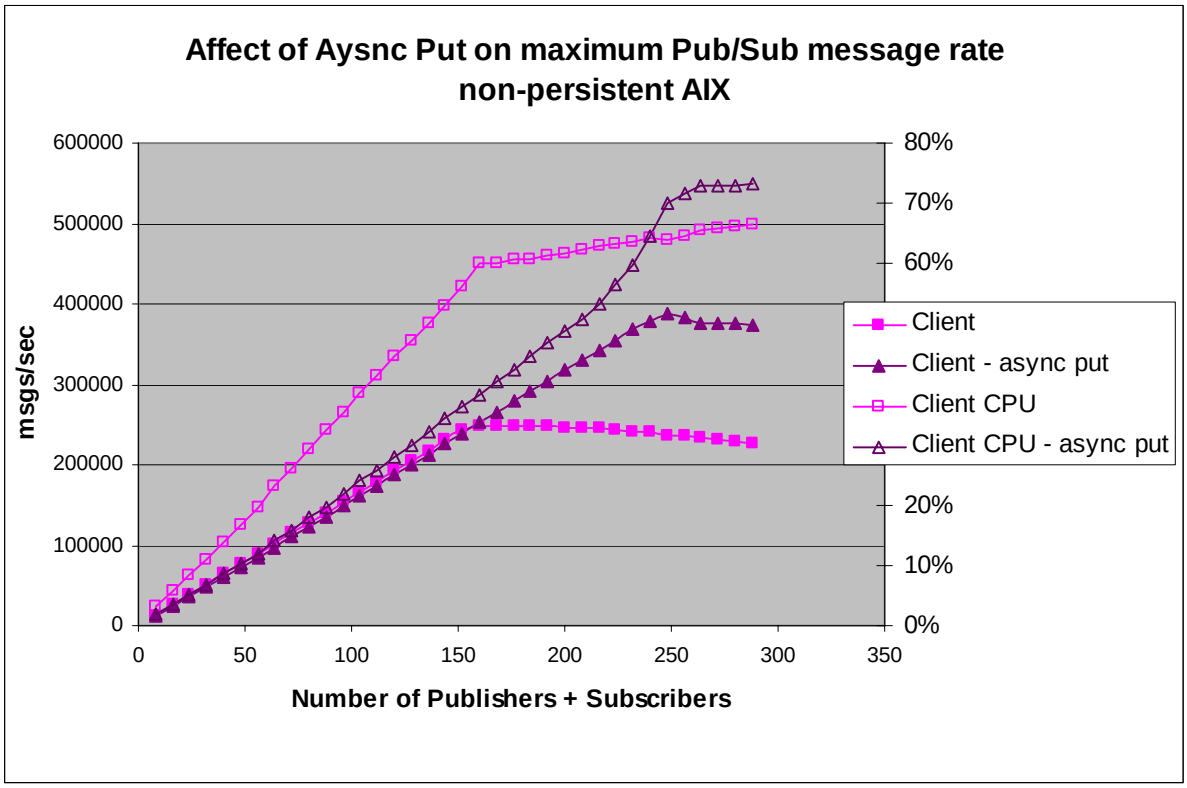


Chart 6 - Affect of AsyncPut on maximum Pub/Sub message rate non-persistent AIX

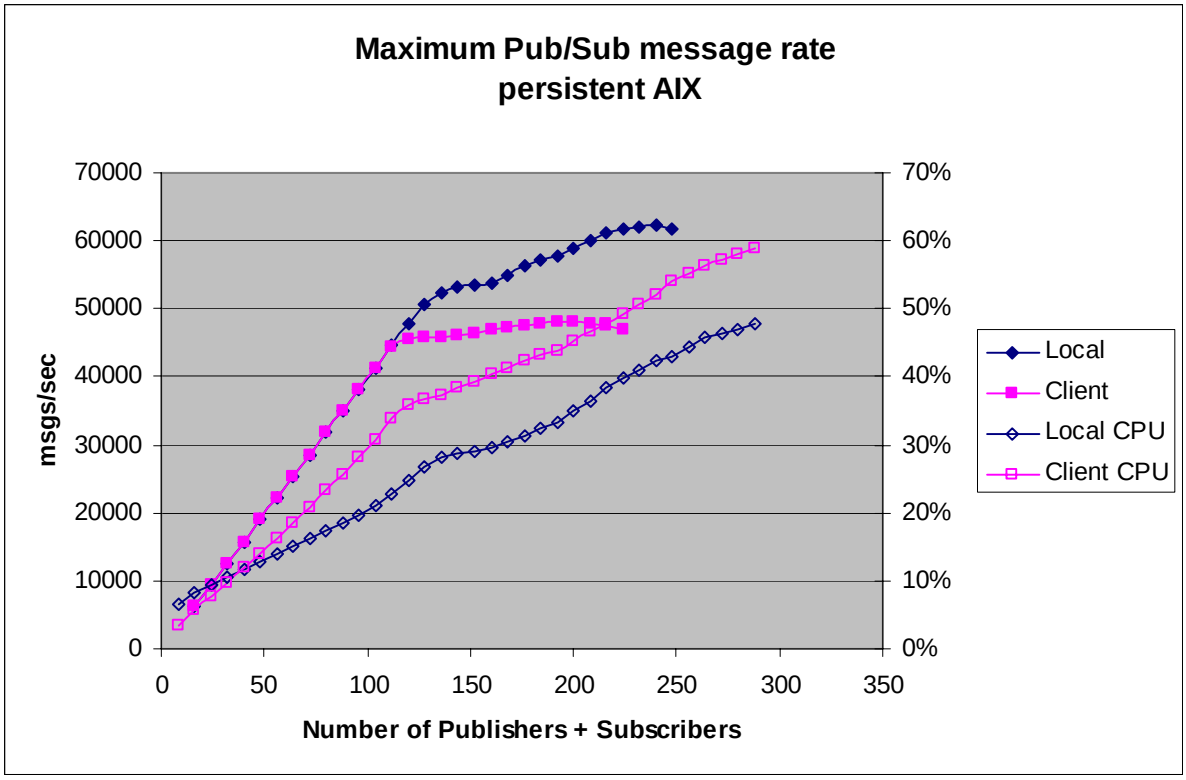


Chart 7 - Maximum Pub/Sub message rate persistent AIX

Chart 21 - Maximum Pub/Sub message rate persistent Linux64
Chart 33 - Maximum Pub/Sub message rate persistent Windows

The maximum throughput in the local persistent scenario is limited by the write speed of the SAN disks used to persist the messages.

2.3 Affect of Publisher fan-out on message rate

The previous two test scenarios show how message rate is affected by increasing the number of subscribers with one publisher or by having one publisher per subscriber and increasing the number of publisher/subscriber pairs. In reality, many customer scenarios will be a combination of these two factors.

Each line on the graph below represents a different number of subscribers per topic with a single publisher. The number of subscribers per topic is referred to as the publisher fan-out. For example a Publisher publishing to a topic with 5 Subscribers is shown in the graphs as Fan5. The number of publishers and associated subscribers is increased gradually during the test run.

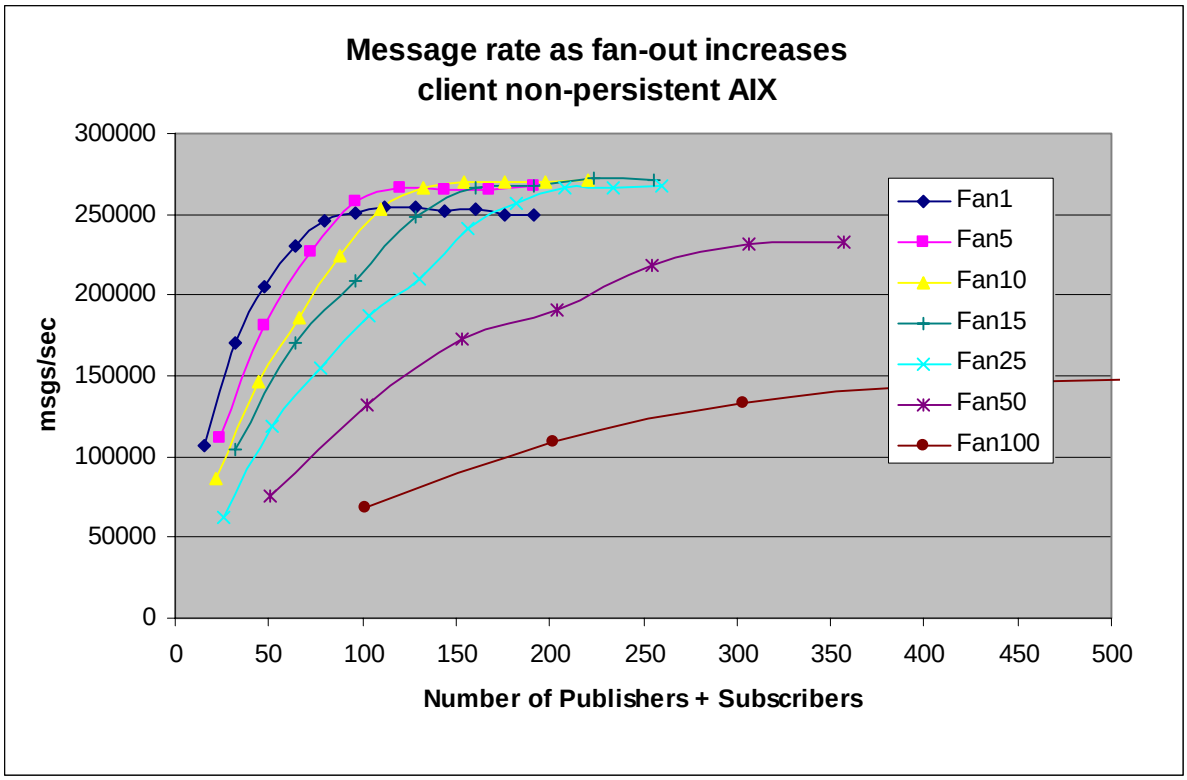


Chart 8 – Message rate as fan-out increases client non-persistent AIX

The results show that for a single publisher on each topic publishing to between 1 and 25 subscribers the maximum message rate is relatively constant. A publisher publishing to more than 25 subscribers will limit the total message rate which can be achieved on the Queue Manager.

2.4 Comparison of Publish/Subscribe and Point-to-Point message rates

Since Local tests give the highest total message rate this is the configuration used for this comparison.
This test scenario compares the maximum message rates achieved with one Publisher and one Subscriber on each Topic with the maximum message rates achieved with one Putter and one Getter on each Queue, this gives the fairest like-for-like comparison of publish/subscribe messaging and point-to-point messaging. Publishers and Putters are referred to as Producers, and Subscribers and Getters as Consumers.
Producers are rated and produce messages at 1600 msgs/sec.

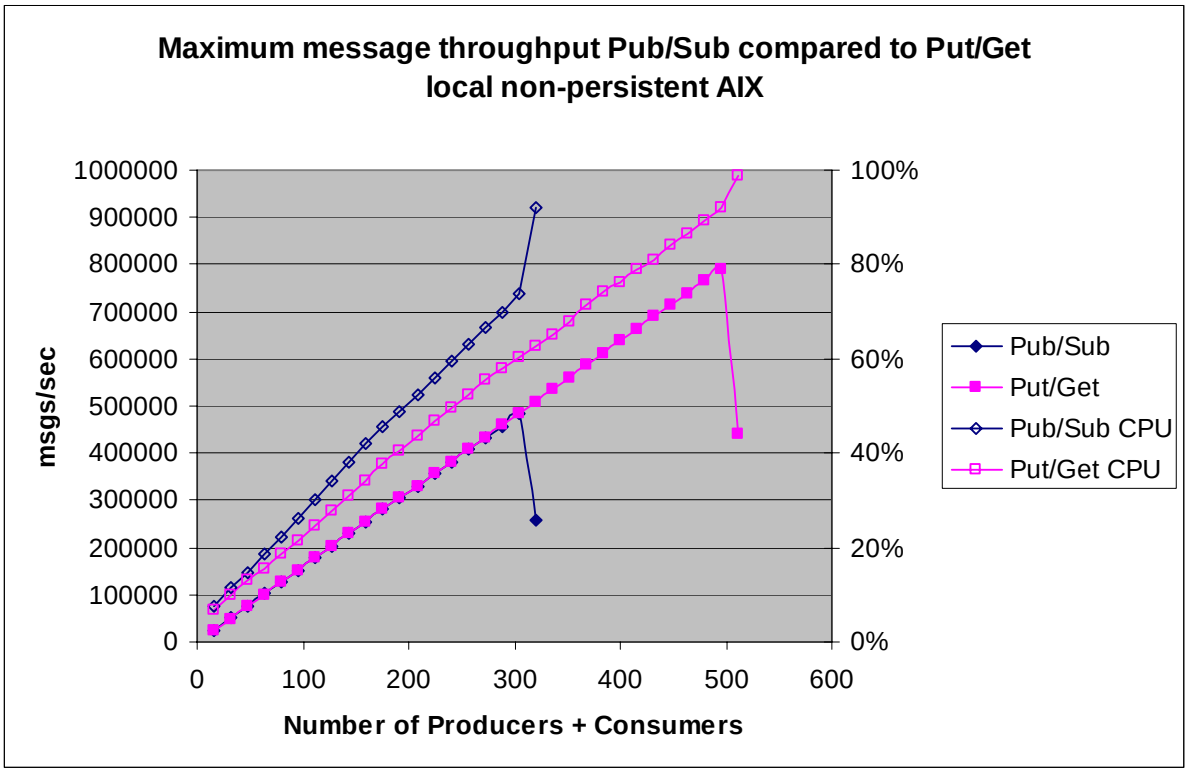


Chart 9 – Maximum message rate Pub/Sub compared to Put/Get local AIX

Chart 22 - Maximum message rate Pub/Sub compared to Put/Get local Linux64
Chart 34 - Maximum message rate Pub/Sub compared to Put/Get local Windows

The results show that Publish/Subscribe uses more CPU per message and hence achieves a lower maximum message rate. It should be noted that some of the CPU is used by the client applications since they are running on the same machine as the Queue Manager. However the same test application was used to drive the Publish/Subscribe and the Put/Get tests and measurements show that it uses a comparable amount of CPU in the two cases.

2.5 Comparison of MQI and JMS

This test scenario compares the maximum message rates achieved with one Publisher and one Subscriber on each Topic using MQI and JMS clients. Publishers are rated and publish messages at 1600 msgs/sec.

Since the JMS client uses significantly more CPU than the MQI test client it is not helpful to compare the maximum throughput using Local tests where the clients run on the same machine as the Queue Manager. This scenario uses Client non-persistent tests to compare the maximum message rate achieved using JMS and MQI clients.

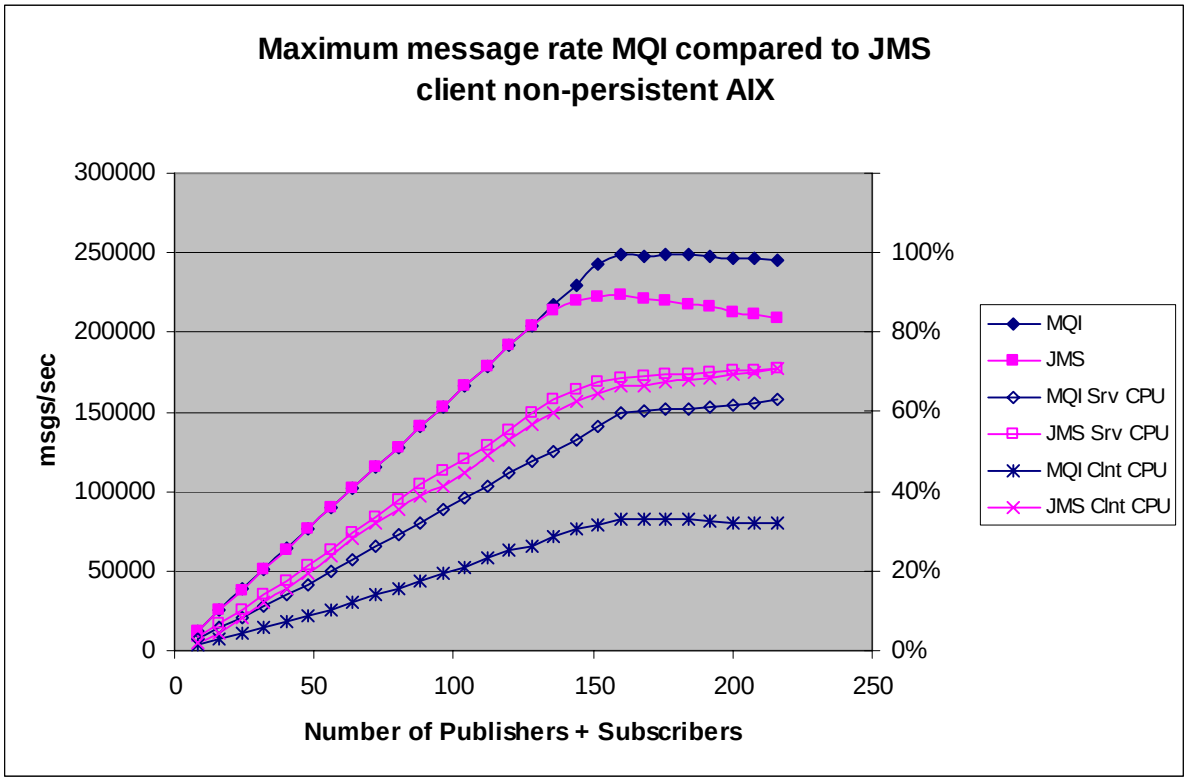


Chart 10 - Maximum message rate MQI compared to JMS client non-persistent AIX

Chart 23 - Maximum message rate MQI compared to JMS client non-persistent Linux64
Chart 35 - Maximum message rate MQI compared to JMS client non-persistent Windows

Operating System	JMS max rate as percentage of MQI max rate
AIX	86%
Linux64	80%
Windows	90%

The results show that the maximum massage rate when processing JMS messages is between 80% and 90% of the maximum message rate when processing MQI messages. The graph also shows that the CPU required to process JMS messages is higher than for MQI messages on both the server and the client machines.

This is expected because JMS message headers are larger and more complex than MQI message headers, which means that there is more data being sent between machines and additional processing required in both the client and the Queue Manager. For this test scenario which uses a 2K message body, the message including the header is about 10% larger for JMS than for MQI. This will vary depending on the number of JMS properties that are set on the message.

2.6 Affect of message size on maximum message rate

This test scenario compares the maximum message rates achieved using MQI clients for a range of message sizes. There is one Publisher and one Subscriber on each topic. The number of publishers, subscribers and topics is gradually increased during the test run until the maximum throughput is reached. Publishers are unrated. The Client test configuration was used for this comparison. This data is intended for capacity planning use.

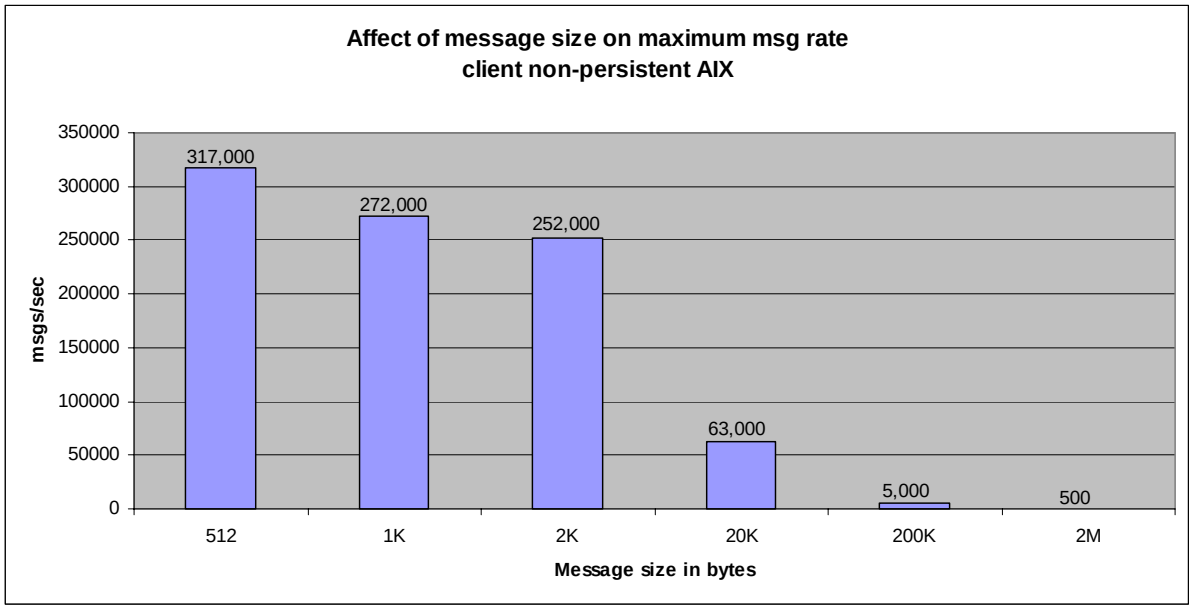


Chart 11 – Affect of message size on message rate client non-persistent AIX

Chart 24 - Affect of message size on message rate client non-persistent Linux64
Chart 36 - Affect of message size on message rate client non-persistent Windows

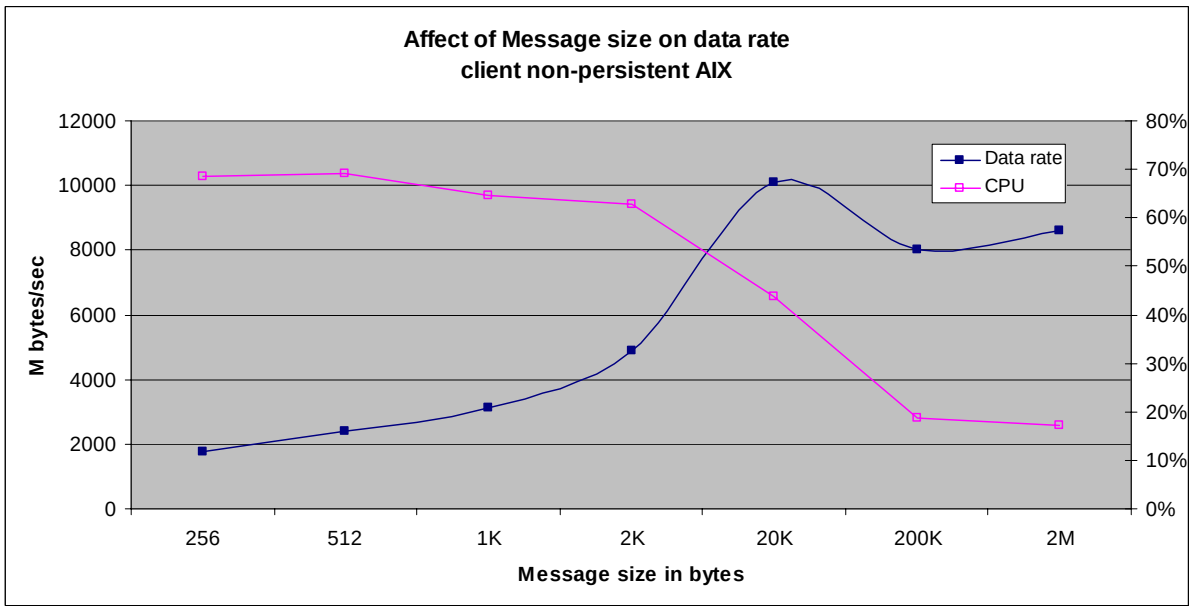


Chart 12 - Affect of message size on data rate client non-persistent AIX

Chart 25 - Affect of message size on data rate client non-persistent Linux64
Chart 37 - Affect of message size on data rate client non-persistent Windows

The results show that for non-persistent messages the maximum data rate is achieved with a 20KB message. It should be noted that the rate includes published messages and messages consumed by the subscribers so the maximum data rate is bi-directional.

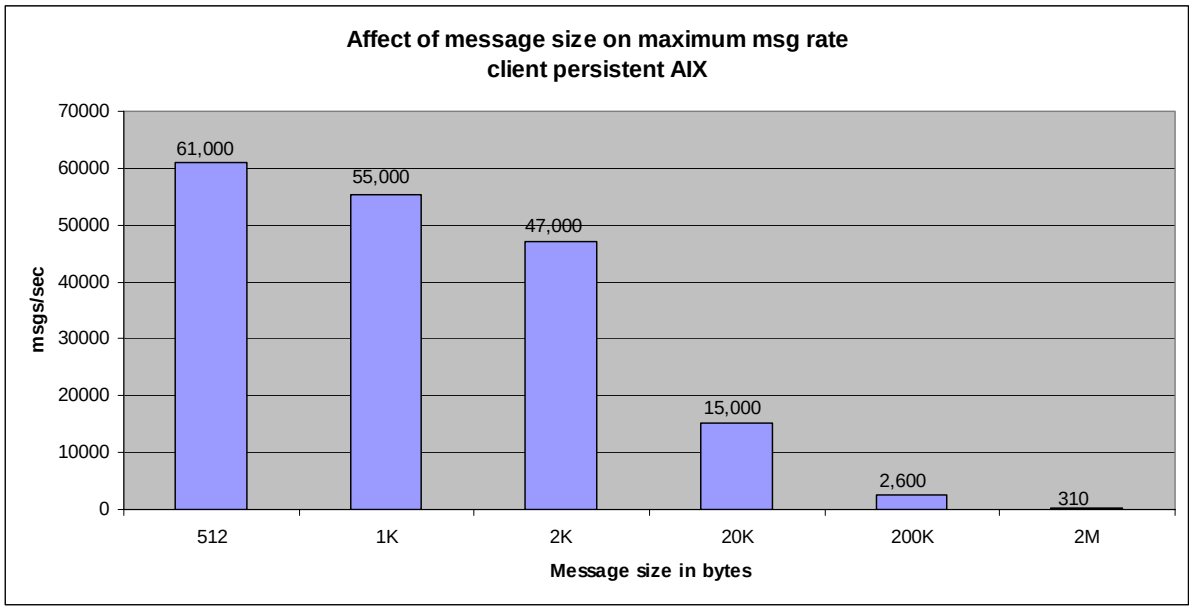


Chart 13 - Affect of message size on message rate client persistent AIX

Chart 26 - Affect of message size on message rate client persistent Linux64
Chart 38 - Affect of message size on message rate client persistent Windows

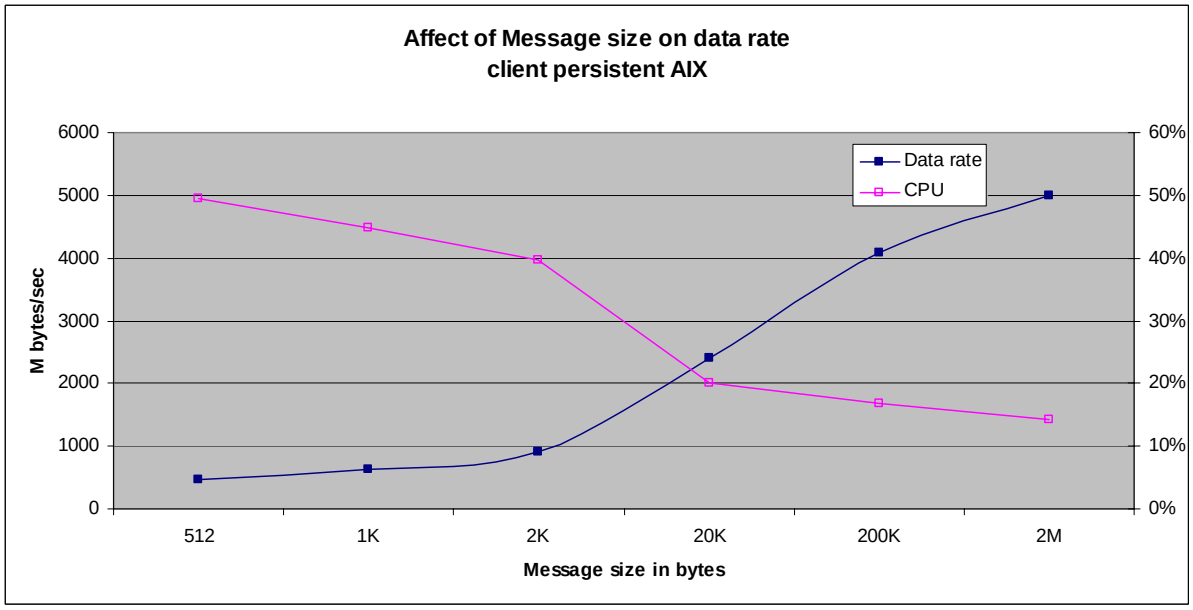


Chart 14 - Affect of message size on data rate client persistent AIX

Chart 27 - Affect of message size on data rate client persistent Linux64
Chart 39 - Affect of message size on data rate client persistent Windows

The results show that for persistent messages the data rate increases as the message size increases. This is because the rate is governed by the speed at which data can be persisted to the SAN and the SAN write time per byte decreases as the message size increases.

2.7 Creating large numbers of Subscriptions

The following section looks at the performance considerations when the rate of subscription creation is important. This might be because you have a large total number of subscriptions to create in a short period of time or because the rate of change of subscriptions in your system is high, for example non-durable subscriptions are continually being created and deleted as applications connect and disconnect. Creating connections is costly in performance terms in comparison to creating subscriptions. For this reason when creating large numbers of subscriptions it is preferable to share connections across multiple subscriptions. This improves the rate at which subscriptions can be created.

The graph below shows the rate at which subscriptions were created with different numbers of subscriptions per connection. The subscriptions created were non-durable.

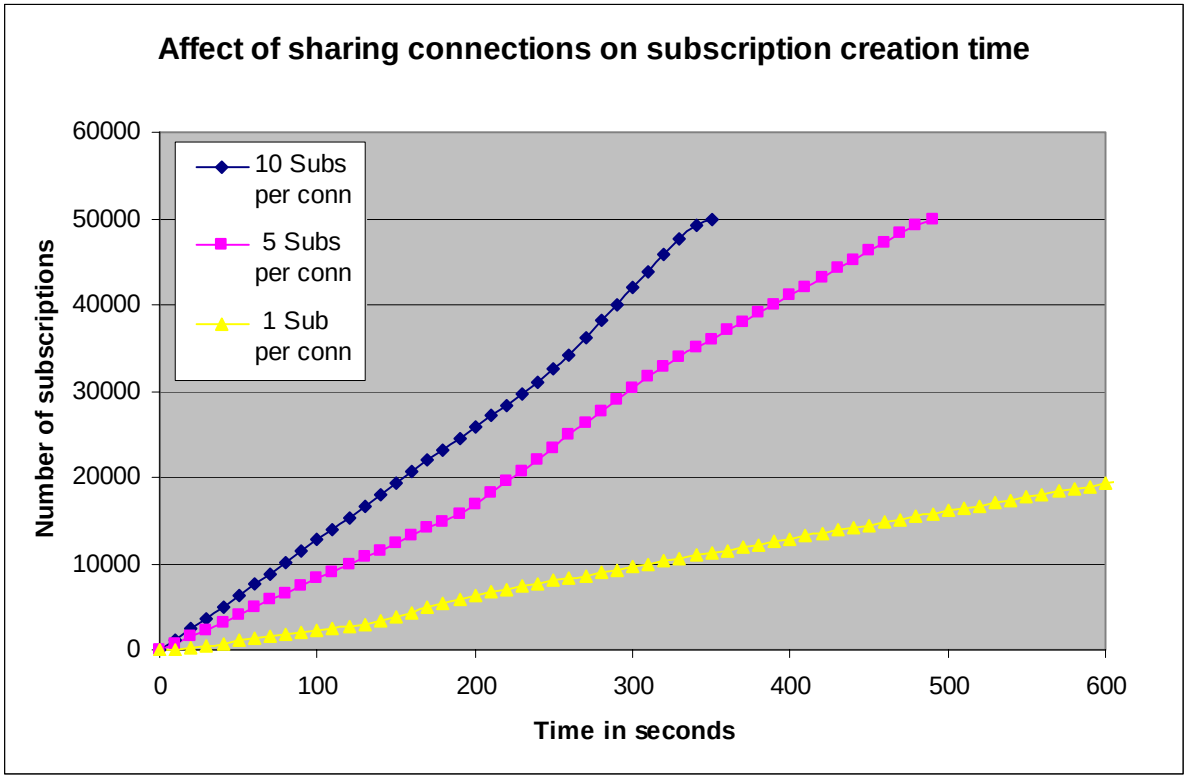


Chart 15 – Affect of sharing connections on subscription creation time

The results show that in 5 minutes it is possible to create:-

- 42,000 subscriptions with 10 subscriptions per connection
- 30,000 subscriptions with 5 subscriptions per connection
- 9,600 subscriptions with 1 subscriptions per connection

Another consideration is the time taken to create the queue associated with each subscription. The subscriptions created in the tests in this report use managed queuing, which means that a subscription queue is automatically created by WebSphere MQ. If the message rate in the system is relatively low and the rate of subscription creation is high then it may be advantageous to not use managed queuing but to create a number of queues and share these across subscriptions.

2.8 Clustering

There are a number of reasons to perform publish/subscribe messaging across a WebSphere MQ cluster of queue managers. Either because the cluster is required to provide messaging across a physically distributed WebSphere MQ infrastructure or to improve availability by providing redundancy across Queue Managers. Alternatively it may be advantageous to use a clustered topology to spread the load of a large publish/subscribe system, even within a single machine. For example, if you have a very high number of subscribers (greater than 100,000 on a large system such as that used in these tests or less on a smaller system) then it may be preferable to split these across a number of Queue Managers. This will reduce the impact of a Queue Manager restart by reducing the time taken to restart and by limiting the impact of a restart to a subset of the subscriptions in the system. It will also simplify the task of administering each individual queue manager.

It is important to understand that there is a performance cost associated with WebSphere MQ clustering due to the requirement to send messages between the Queue Managers in order to reach their target. This is true when using publish/subscribe and WebSphere MQ clusters, when publishers are connected to different queue managers from where the subscribers are connected.

To create a Publish/Subscribe cluster you configure administrative topic objects and define the cluster attribute (WebSphere MQ v7.5 Information Center - Cluster Topics). Information about each clustered topic object is sent to all Queue Managers in a cluster. When the first application subscribes to a topic string at or below a clustered topic in the topic tree a proxy subscription for that topic string is created and sent from the Queue Manager where the subscription is created, to all the other members of the cluster for that subscriber's topic string.

This means that there are additional messages flowing between Queue Managers when subscriptions are created and deleted on different topic strings. This can affect the maximum publication rate which can be sustained especially if there is a high rate of change of subscriptions across many different topic strings.

Messages published on a topic are sent to every matching subscription known to the queue manager that the publisher is connected to. If any of those subscriptions are proxy subscriptions, a single copy of the published message is sent to the queue manager that originated the proxy subscription. The receiving queue manager then sends a copy of the message to every local subscription matching that topic. This ensures that the subscriber to a clustered topic receives publications from publishers connected to any of the queue managers in the cluster.

This means that additional messages are sent from each Queue Manager that has publishers connected to it to other Queue Managers in the cluster where subscriptions exist. If most messages published need to be sent to subscribers on many Queue Managers in the cluster then this will limit the maximum publication rate achievable.

The cluster test in this section of the report shows the performance implications of using 2 and 4 Queue Manager clusters. **All Queue Managers in the tests were run on the same machine.**

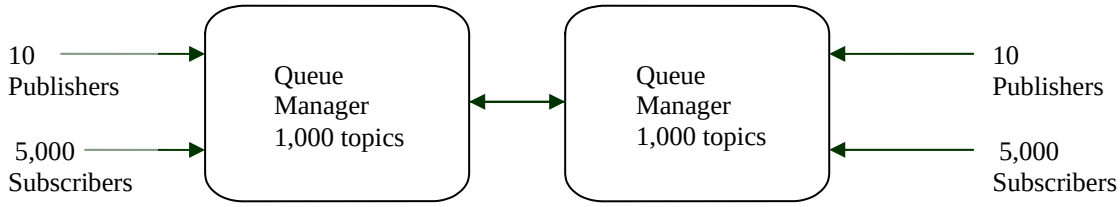
- The test uses 10,000 subscribers on 1,000 topics, which means that there are 10 subscriptions matching each topic and so every message published is delivered to 10 subscribers.
- To ensure that the publication rate is not limited by the publishers, 20 publishers were used.
- Each publisher publishes a message to each of the 1,000 topics in turn.
- The subscriptions were all created before the publishers were started.
- The message size used for the test is 1KB.
- JMS subscribers and publishers were used.
- Subscribers and publishers were run on separate machines from the WebSphere MQ Queue Manager
- Subscriptions created were non-persistent non-durable

In this test the publishers and subscribers are spread across all queue managers. This results in every publication being distributed to all queue managers in the cluster. Whilst evenly spreading the load across the cluster and reducing any single point of failure it does introduce the maximum messaging overhead on the cluster.

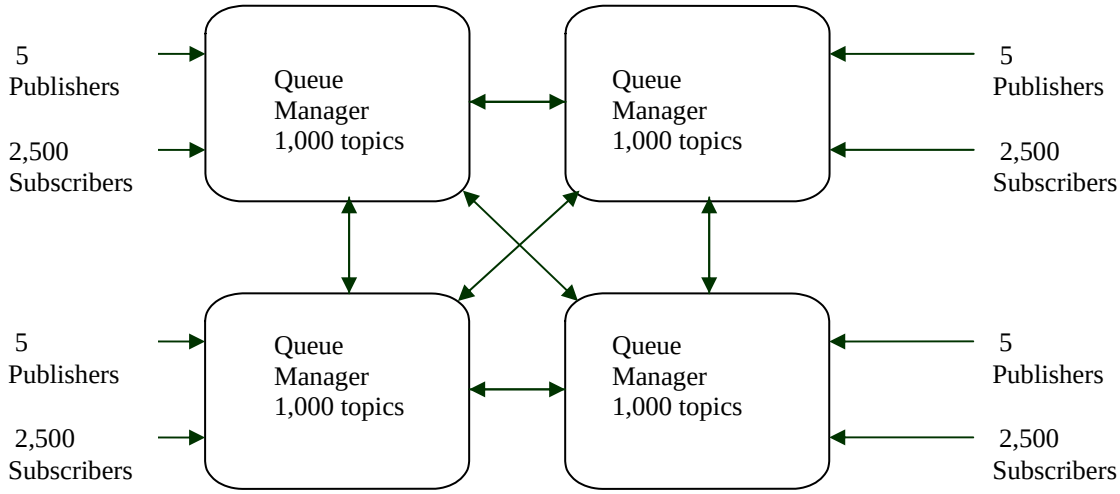
Below are diagrams showing the test configurations for the tests.

Single Queue Manager

Two Queue Manager cluster



Four Queue Manager cluster



The CPU used for each of the Queue Manager configurations above was measured with a steady publication rate of 8,000 msgs/sec across all publishers on all queue managers. Since every message published is delivered to 10 Subscribers the total application message rate is 88,000 msgs/sec. The table below shows the results.

Test configuration	Total CPU at 8,000 publications/sec
1 Queue Manager	32%
2 Queue Manager cluster	38%
4 Queue Manager cluster	49%

This shows the additional performance cost associated with spreading the clients across multiple Queue Managers.

2.9 Tuning a Pub/Sub Cluster

The other measurement of interest is the maximum publication rate which can be achieved by 4 Queue Manager cluster with all Queue Managers running on the same machine.

Test configuration	Publications/sec	CPU
4 Queue Manager cluster	8,000	49%
4 Queue Manager cluster with tuning explained below	15,000	86%

For clustered scenarios the maximum publication rate is limited by the rate at which publication messages can be processed when sent between Queue Managers.

In a Pub/Sub cluster there are two queues on each queue manager that are used to handle messages sent between Queue Managers in the cluster :-

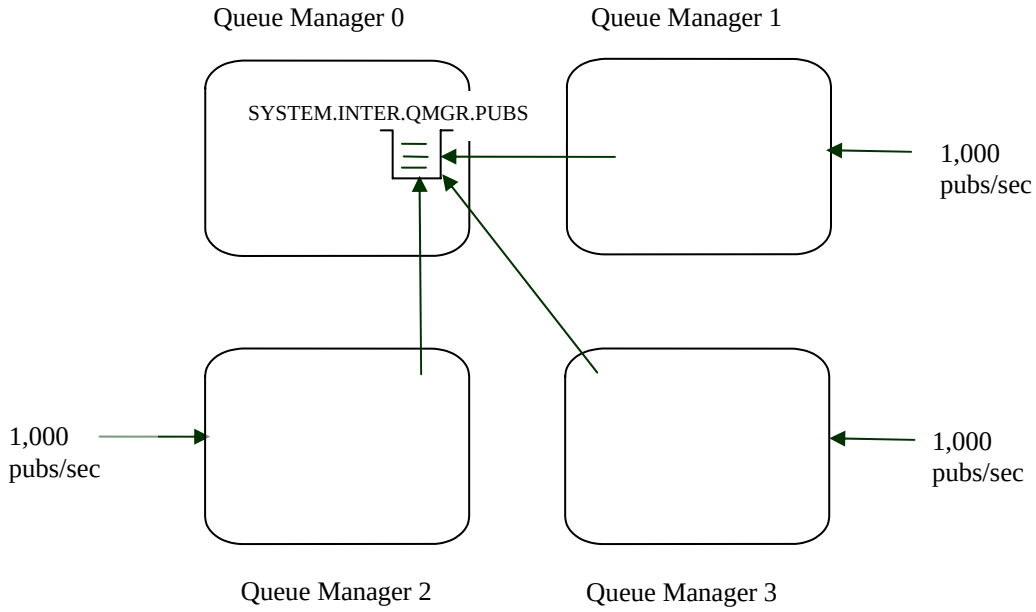
- **SYSTEM.INTER.QMGR.PUBS**
  - **The queue to which publications are sent by each remote queue manager in the cluster. Messages arriving on this queue are published to the local subscribers on this queue manager.**
- **SYSTEM.CLUSTER.TRANSMIT.QUEUE**
  - **The transmission queue used by a queue manager to send publications to each remote queue manager.**

If the cluster is not able to handle the required publication rate and messages are seen to be building up on one of these queues then there is tuning available in WMQ V7.5 which may help. These are explained below.

SYSTEM.INTER.QMGR.PUBS Queue

In the 4 Queue Manager cluster, because there are Subscribers on every topic on each Queue Manager in the cluster every queue manager will receive every message published to a remote queue manager. So for example if the publication rate at each Queue Manager is 1,000 msgs/sec then the rate at which messages arrive on the SYSTEM.INTER.QMGR.PUBS queue on each Queue Manager is 3,000 msgs/sec (1,000 msgs/sec arriving from each remote queue manager).

As the publication rate across the cluster increases messages can build up on the SYSTEM.INTER.QMGR.PUBS queue on each Queue Manager in the cluster if they arrive faster than they can be processed by the queue manager.



Before any tuning was applied to the 4 Queue Manager cluster the maximum sustainable publication rate was **8,000 msgs/sec**. This represents a publications rate of 2,000 msgs/sec on each Queue Manager, which is 6,000 msgs/sec at each SYSTEM.INTER.QMGR.PUBS queue. Increasing the publication rate above this limit caused messages to build up on the SYSTEM.INTER.QMGR.PUBS queue on each Queue Manager.

A tuning parameter is available in WebSphere MQ V7.5 to relieve this bottleneck. This parameter defines the number of threads used to get messages from the SYSTEM.INTER.QMGR.PUBS queue. The default number of threads used is one, this enables message order to be maintained for incoming publications but does restrict the rate at which they can be processed. Increasing the number of threads improves the processing capability but does result in messages being delivered in a different order to them being published, even within a single topic.

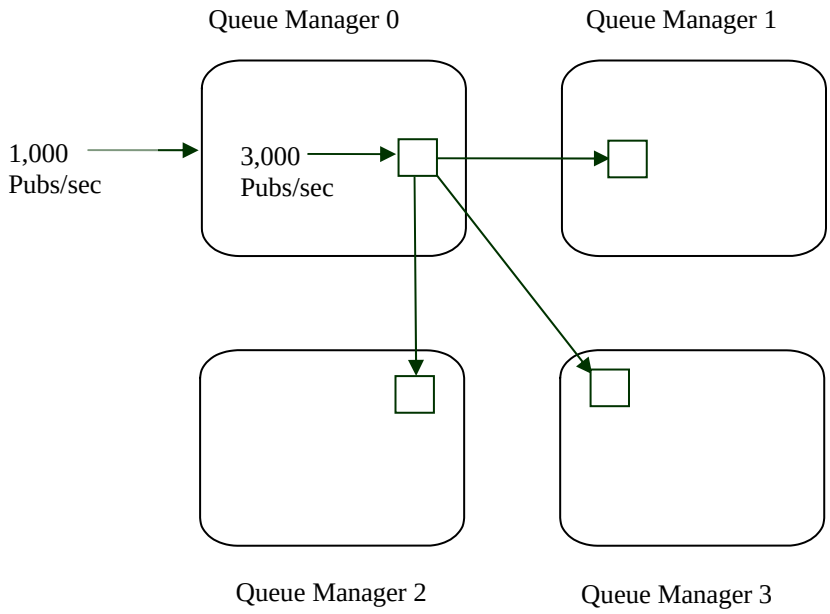
In the tuned test this parameter was set in the qm.ini file for each of the Queue Managers in the cluster :-
TuningParameters:
pscNumPubThreads=5

Setting this to use 5 threads to process messages from each SYSTEM.INTER.QMGR.PUBS queue increased the maximum sustainable publication rate to **12,000 msgs/sec**.

SYSTEM.CLUSTER.TRANSMIT.QUEUE queue

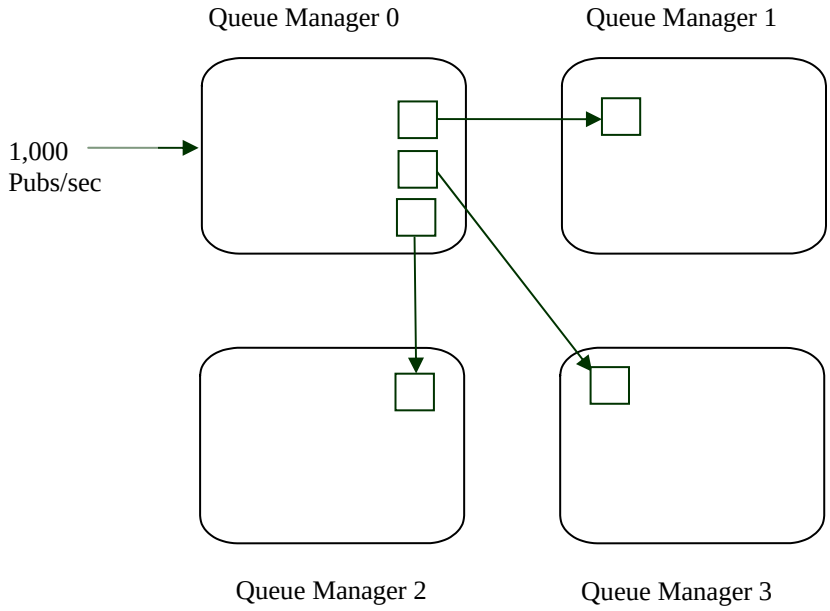
In the 4 Queue Manager cluster, publication messages published by a publisher connected to Queue Manager 0 are delivered to the local subscribers and a message is put onto the SYSTEM.CLUSTER.TRANSMIT.QUEUE for each of the other Queue Managers in the cluster with a matching proxy subscription, in this case all 3. So for example if the publication rate on Queue Manager 0 is 1,000 msgs/sec then the rate at which messages are put onto the SYSTEM.CLUSTER.TRANSMIT.QUEUE is 3,000 msgs/sec. The three cluster sender channels are concurrently taking those messages and sending them to the remote queue managers.

As the publication rate increases messages can build up on the SYSTEM.CLUSTER.TRANSMIT.QUEUE on each Queue Manager in the cluster.



Before any tuning was applied to the SYSTEM.CLUSTER.TRANSMIT.QUEUE the maximum sustainable publication rate was **12,000 msgs/sec**. This represents a publication rate of 3,000 msgs/sec on each Queue Manager, which is 9,000 msgs/sec on the SYSTEM.CLUSTER.TRANSMIT.QUEUE. Increasing the publication rate above this limit caused messages to build up on the SYSTEM.CLUSTER.TRANSMIT.QUEUE on each Queue Manager.

Configuration is available in WebSphere MQ V7.5 to relieve this bottleneck. Multiple cluster transmission queues (WebSphere MQ v7.5 Information Center - Multiple cluster transmission queues) enable you to assign a different cluster transmission queue to each cluster sender-channel. This means that each pair of Queue Managers in the cluster can use a separate transmission queue as shown below.



With the queue manager configured to use a separate transmission queue per cluster channel the maximum sustainable publication rate was increased to **15,000 msgs/sec**

3 Linux64 results

3.1 Affect of number of subscriptions on publication rate

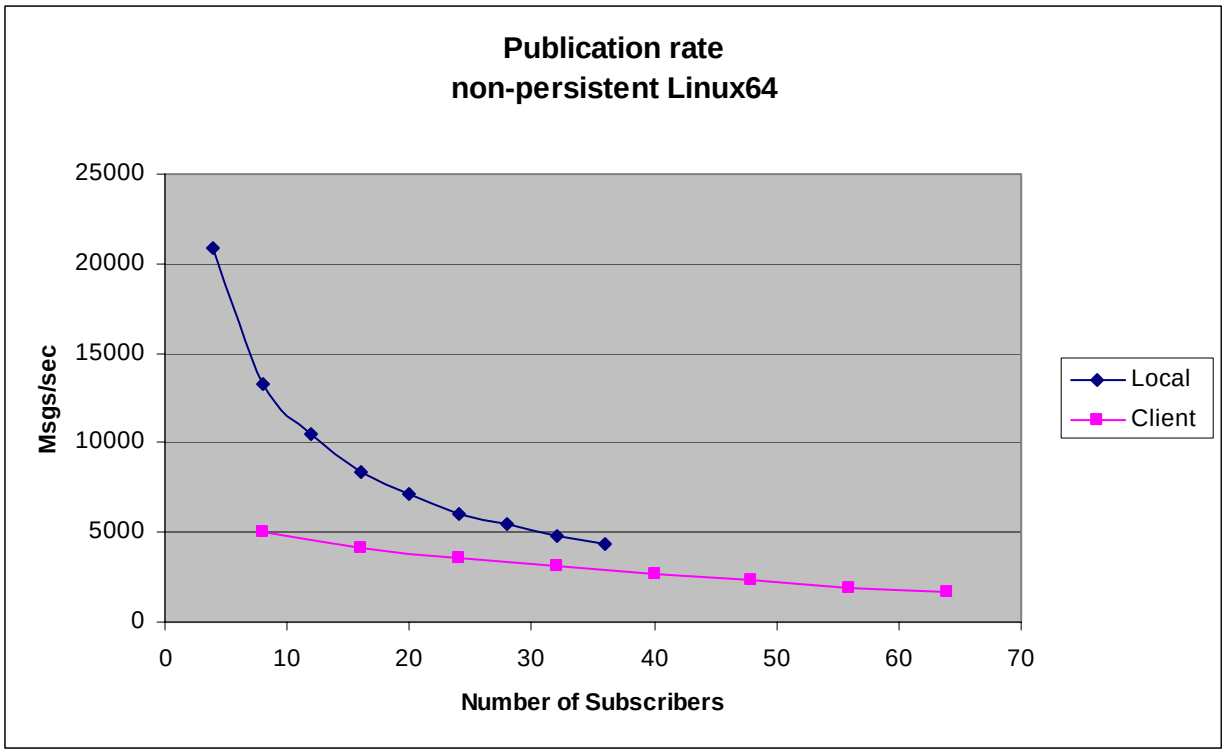


Chart 16 - Publication rate non-persistent Linux64

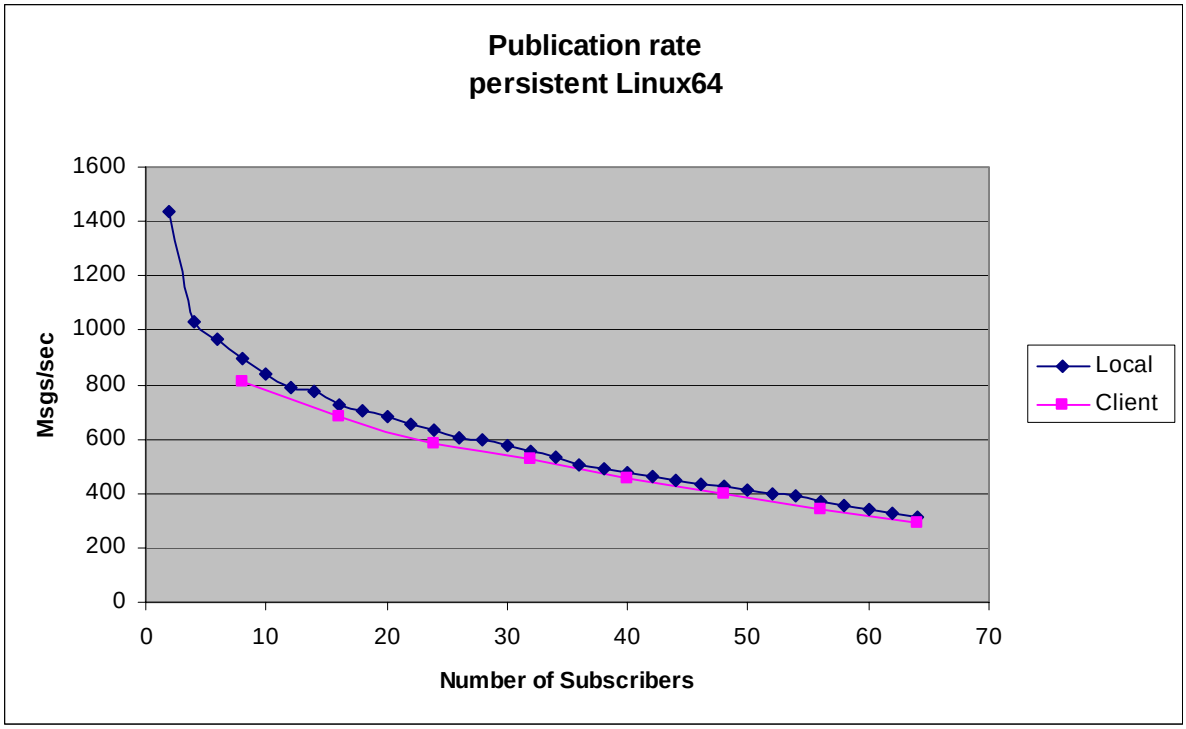


Chart 17 - Publication rate persistent Linux64

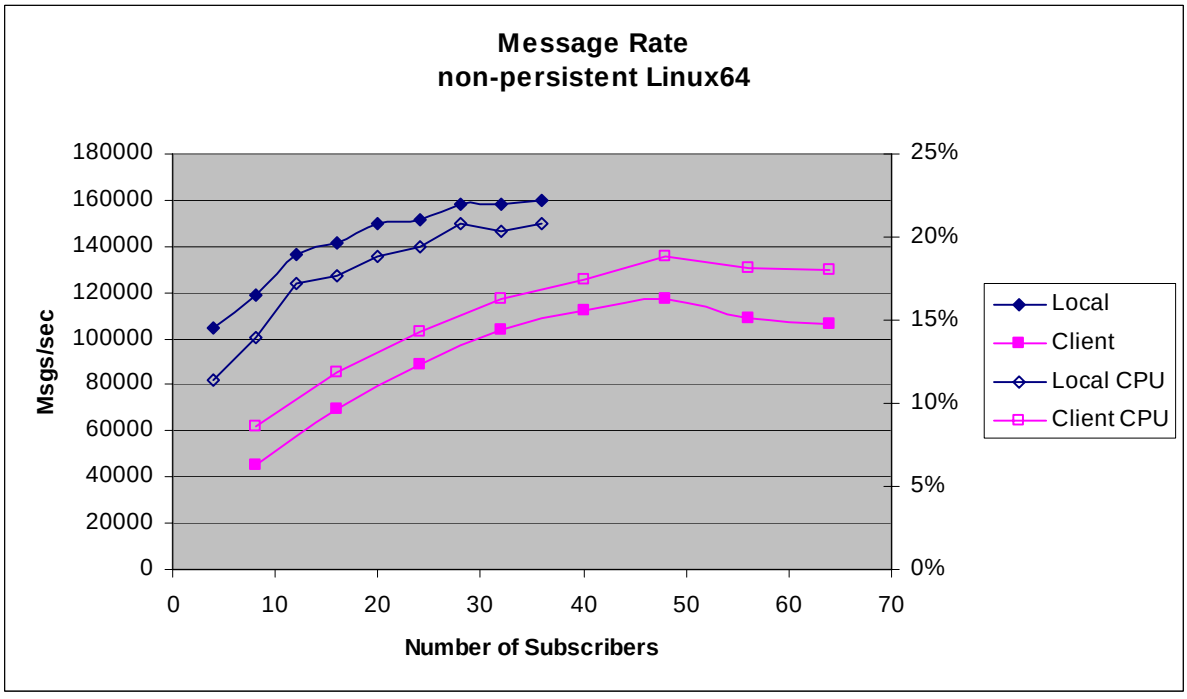


Chart 18 - Message rate non-persistent Linux64

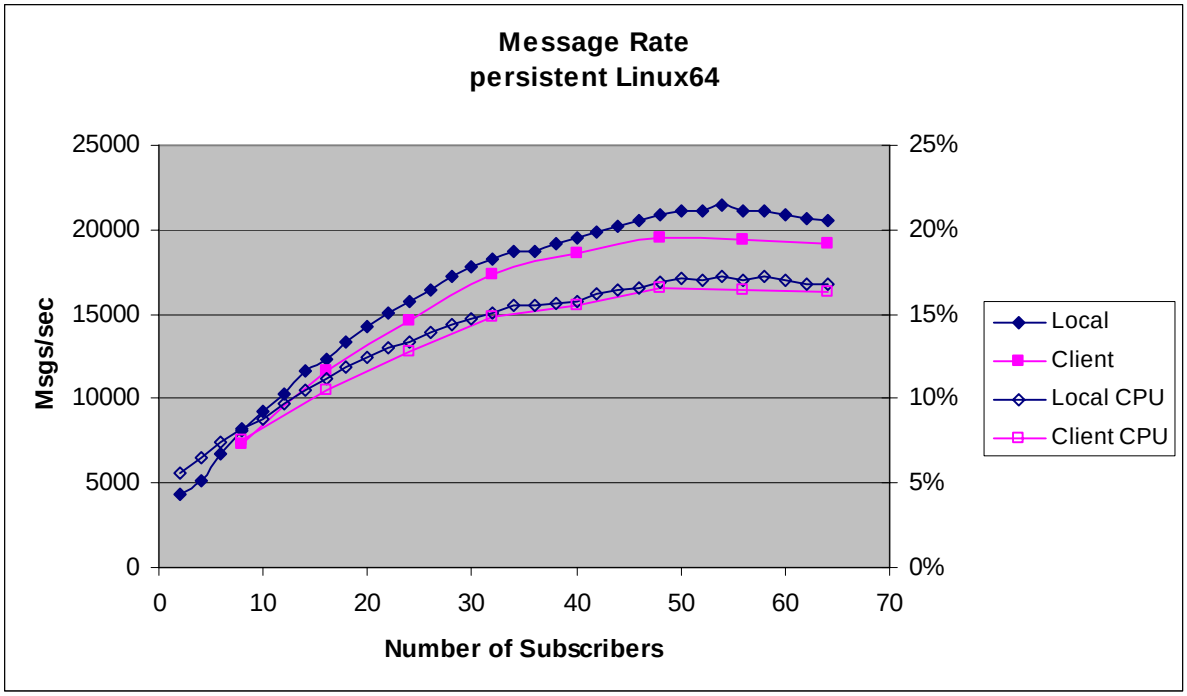


Chart 19 – Message rate persistent Linux64

3.2 Maximum Publish/Subscribe message rate

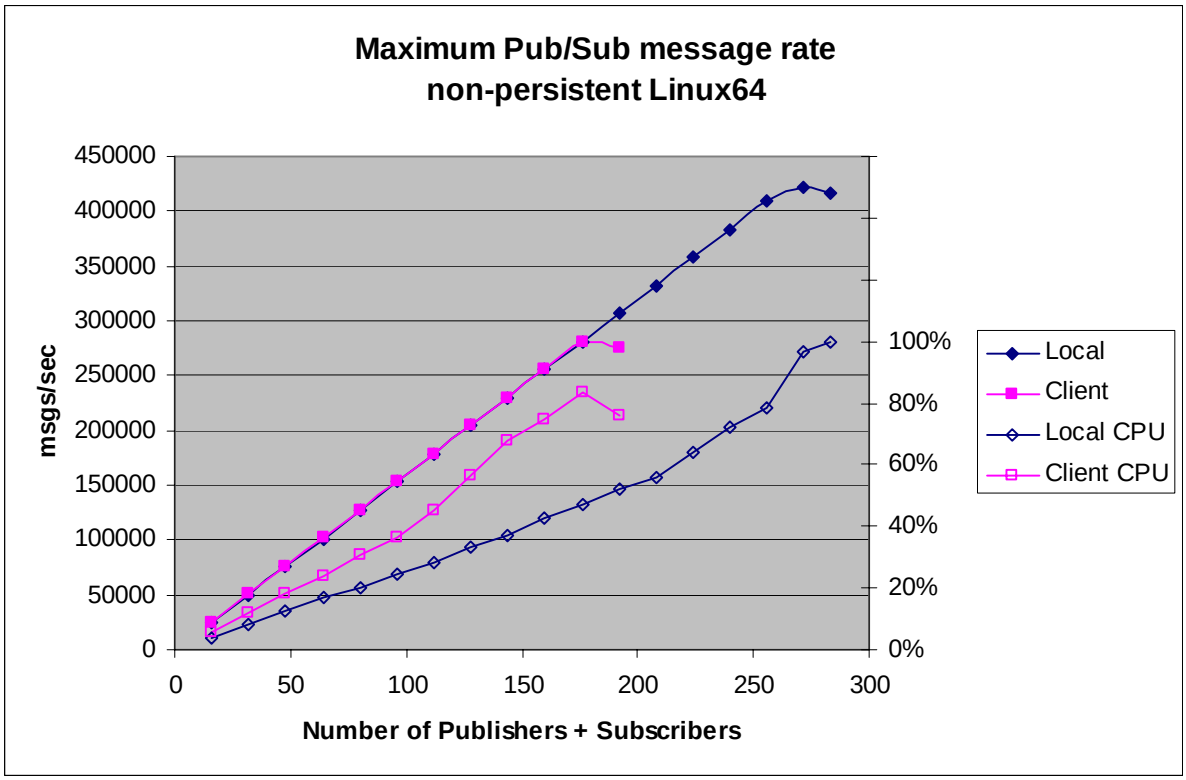


Chart 20 - Maximum Pub/Sub message rate non-persistent Linux64

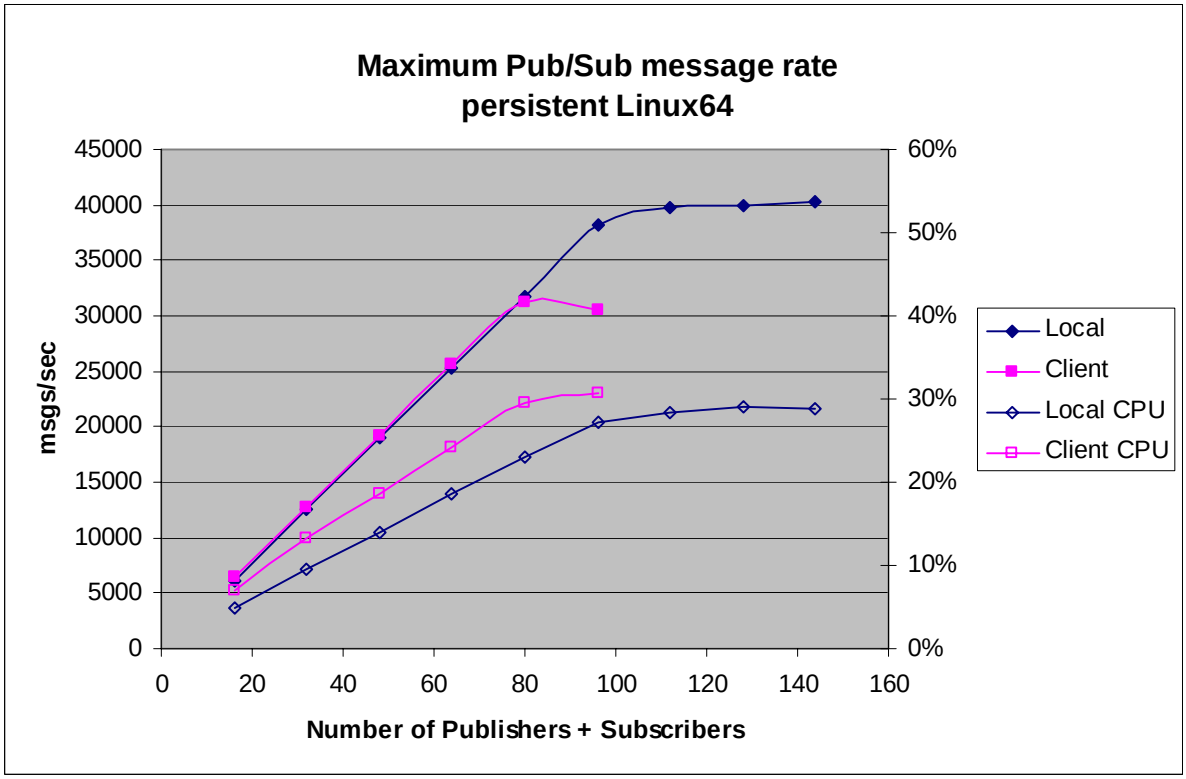


Chart 21 - Maximum Pub/Sub message rate persistent Linux64

3.3 Comparison of Publish/Subscribe and Point-to-Point message rates

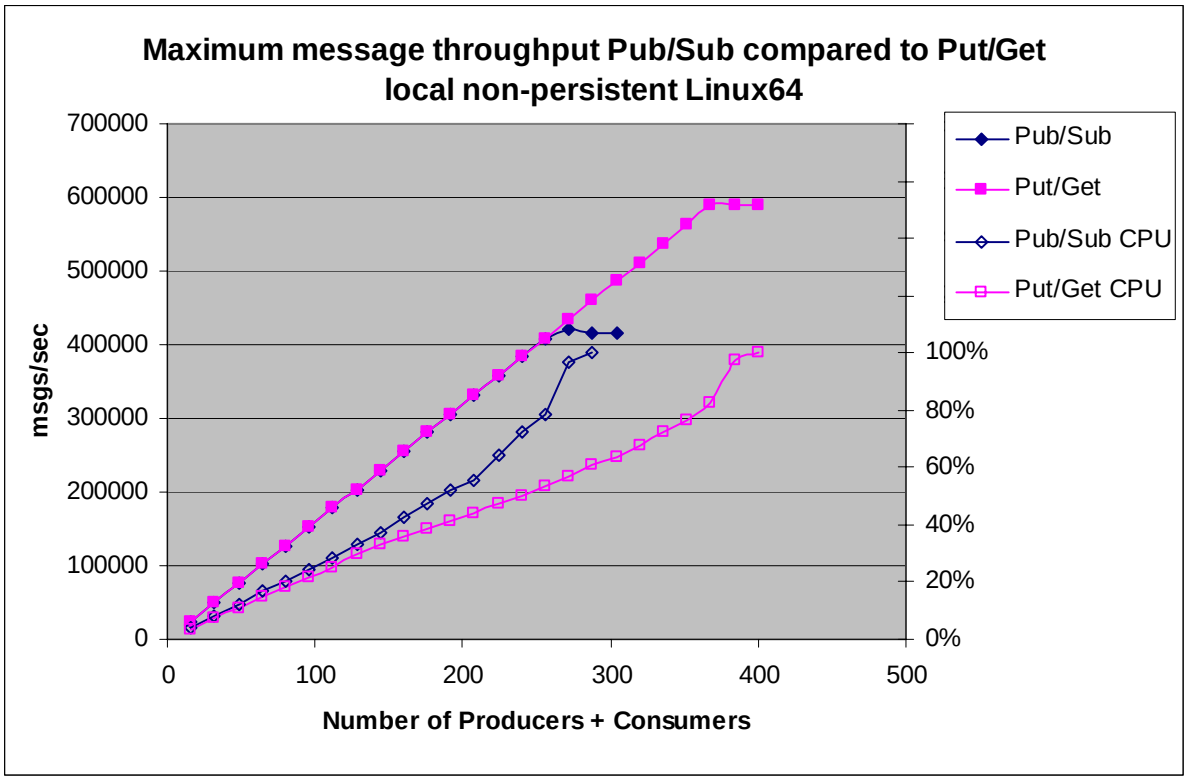


Chart 22 - Maximum message rate Pub/Sub compared to Put/Get local Linux64

3.4 Comparison of MQI and JMS

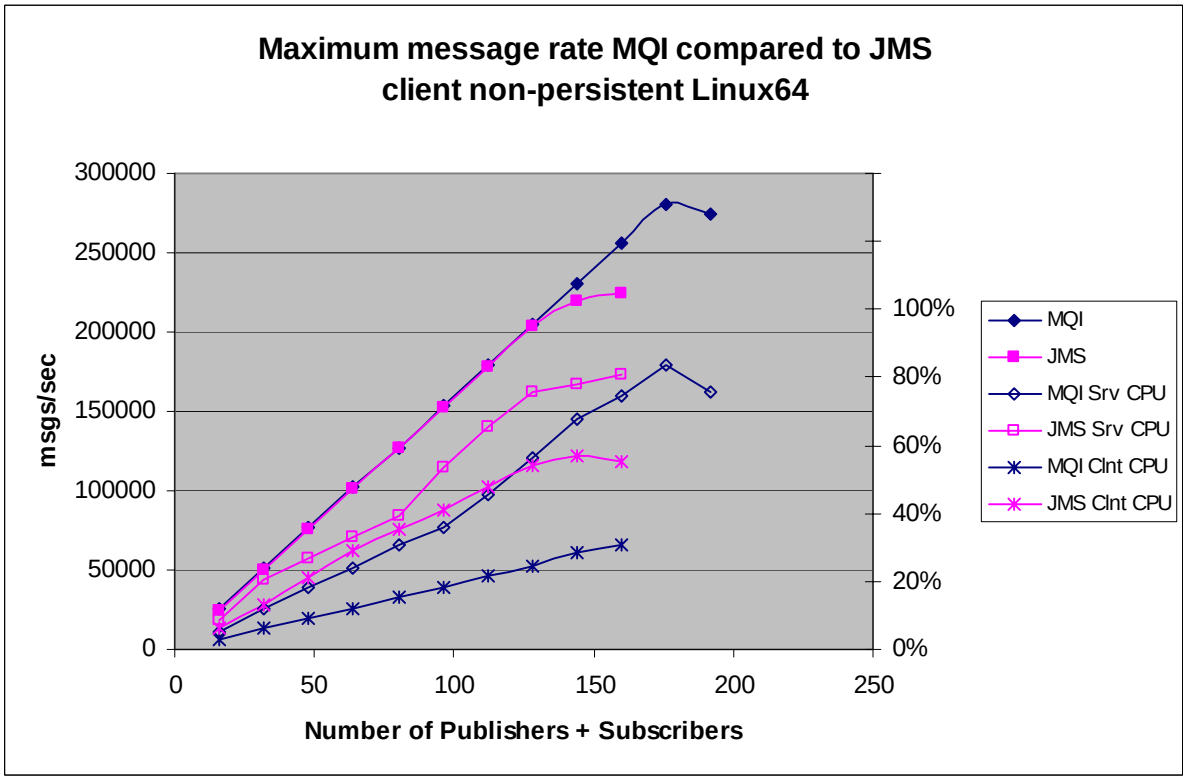


Chart 23 - Maximum message rate MQI compared to JMS client non-persistent Linux64

3.5 Affect of message size on maximum message rate

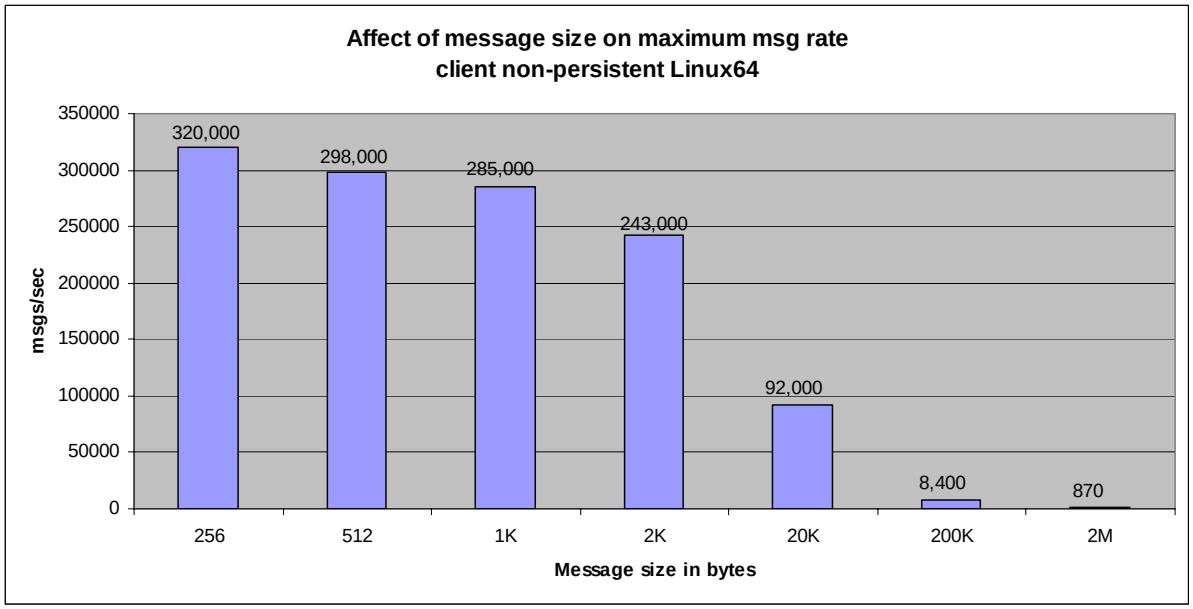


Chart 24 - Affect of message size on message rate client non-persistent Linux64

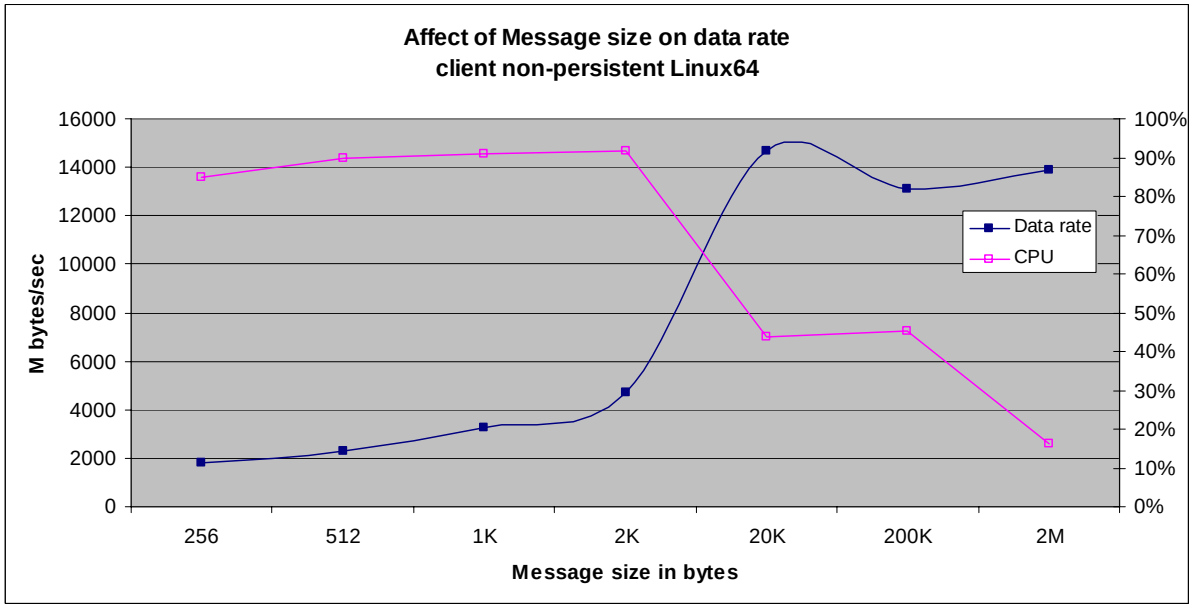


Chart 25 - Affect of message size on data rate client non-persistent Linux64

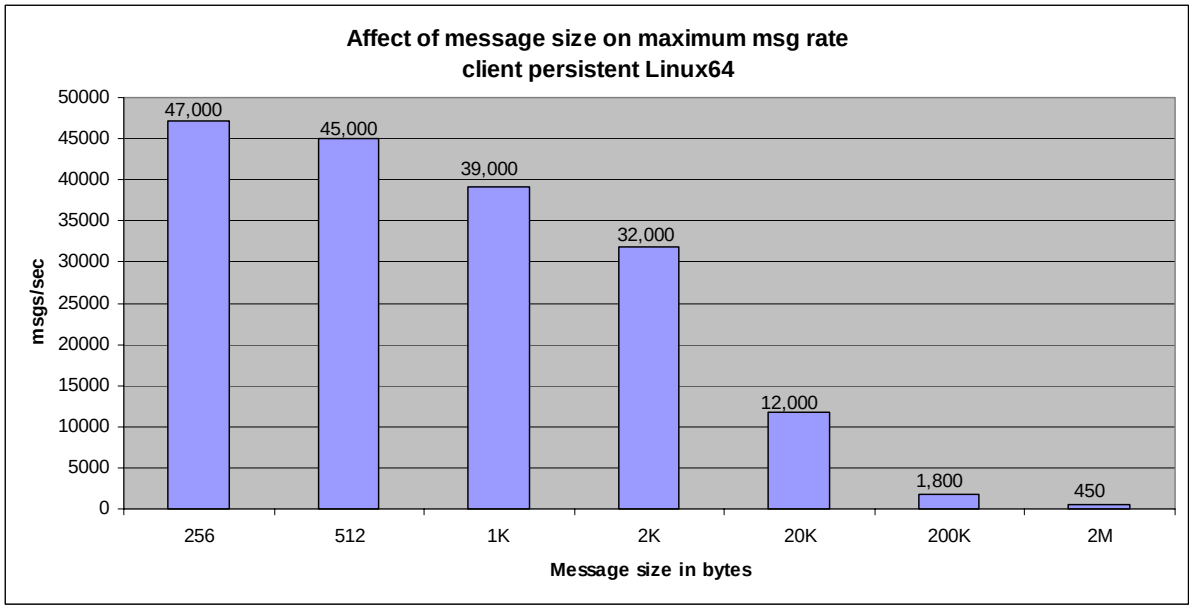


Chart 26 - Affect of message size on message rate client persistent Linux64

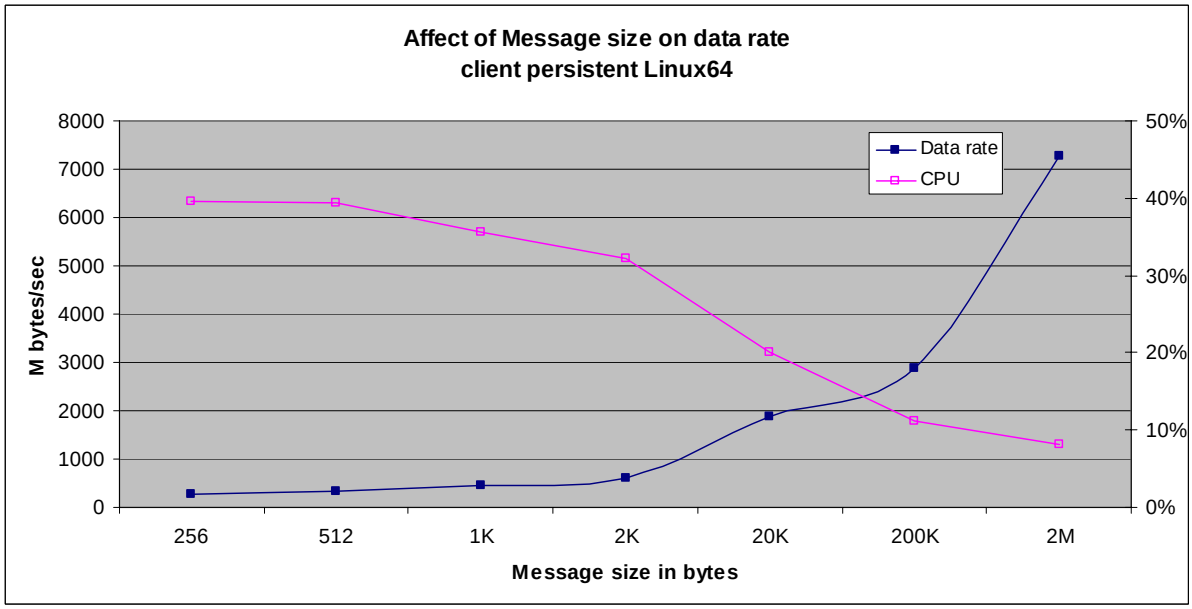


Chart 27 - Affect of message size on data rate client persistent Linux64

4 Windows results

4.1 Affect of number of subscriptions on publication rate

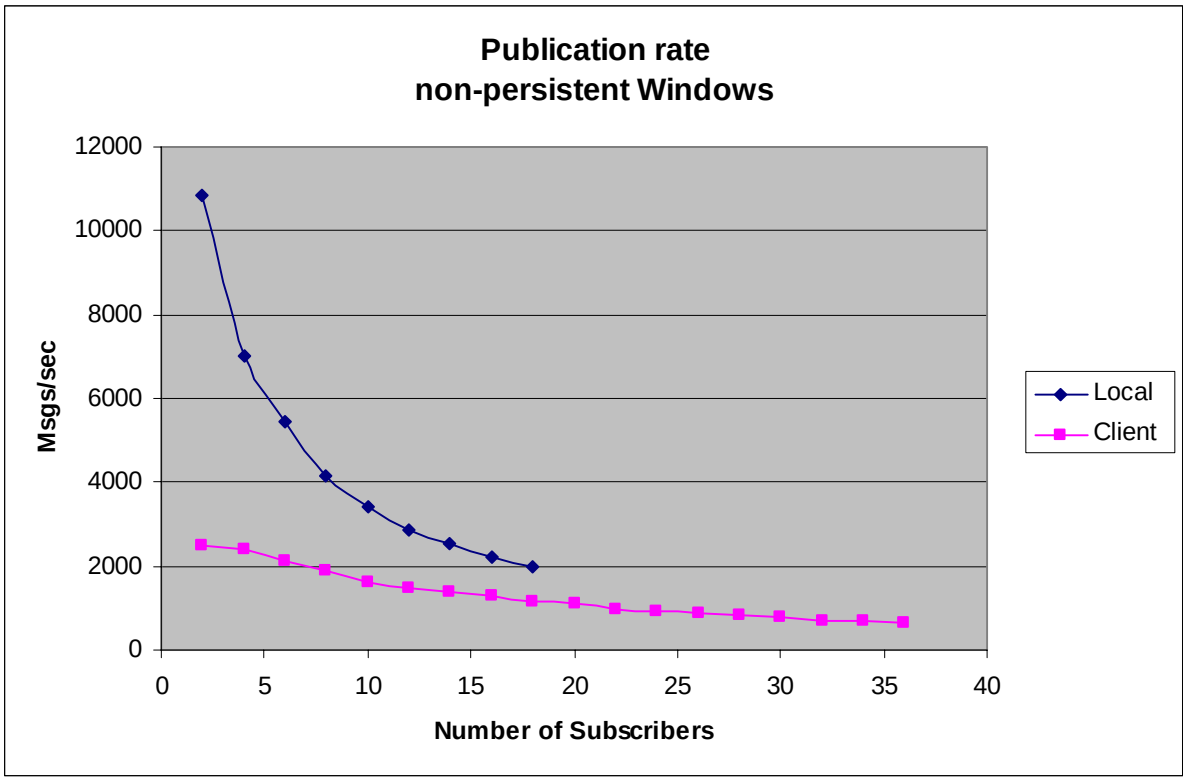


Chart 28 - Publication rate non-persistent Windows

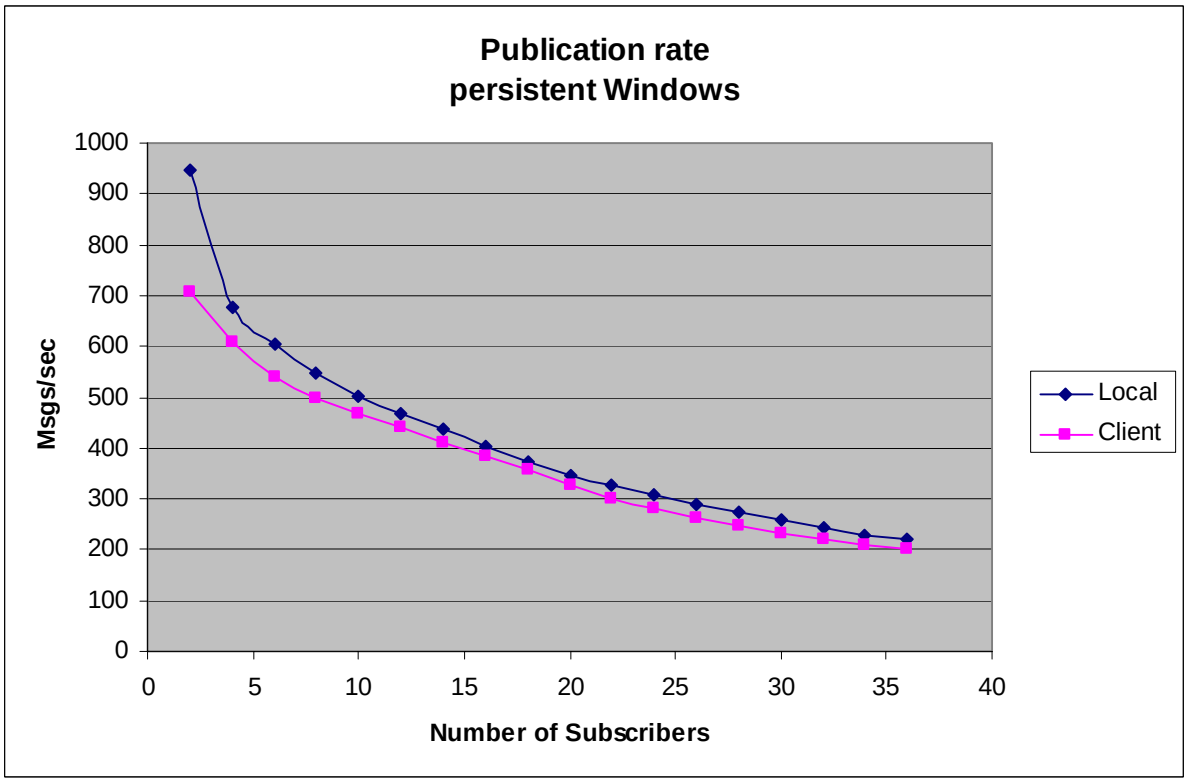


Chart 29 - Publication rate persistent Windows

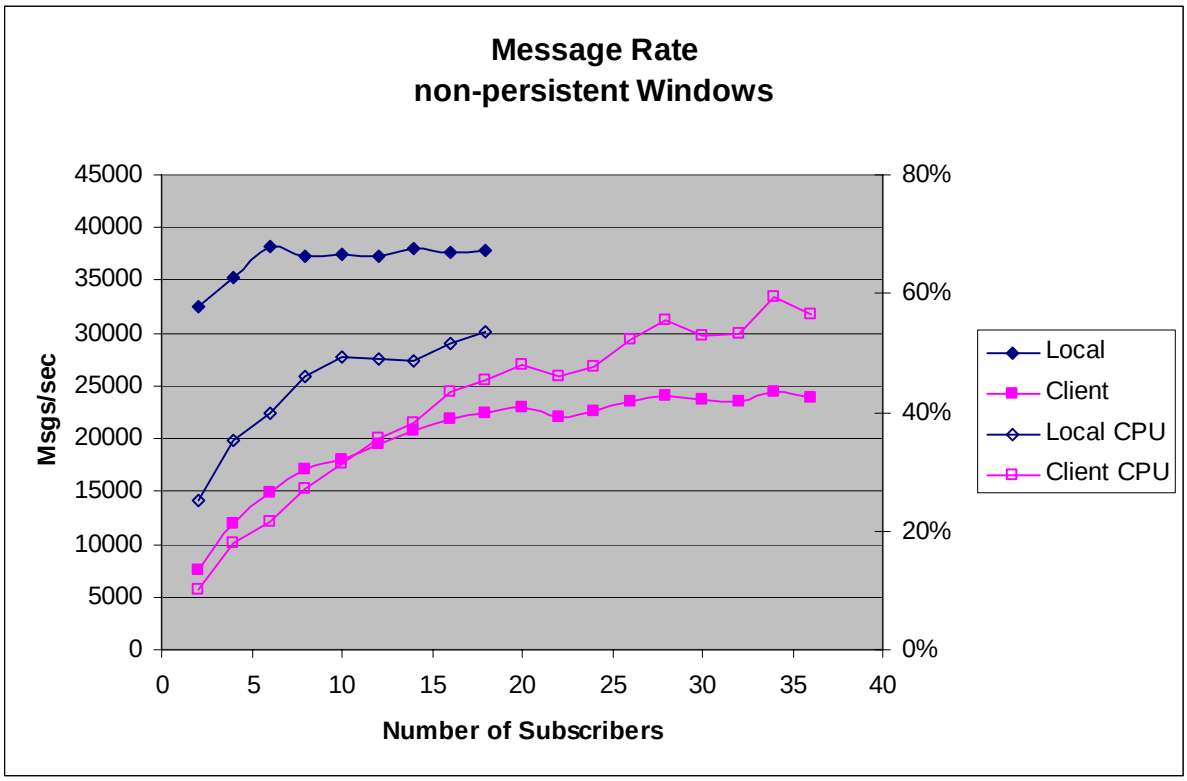


Chart 30 - Message rate non-persistent Windows

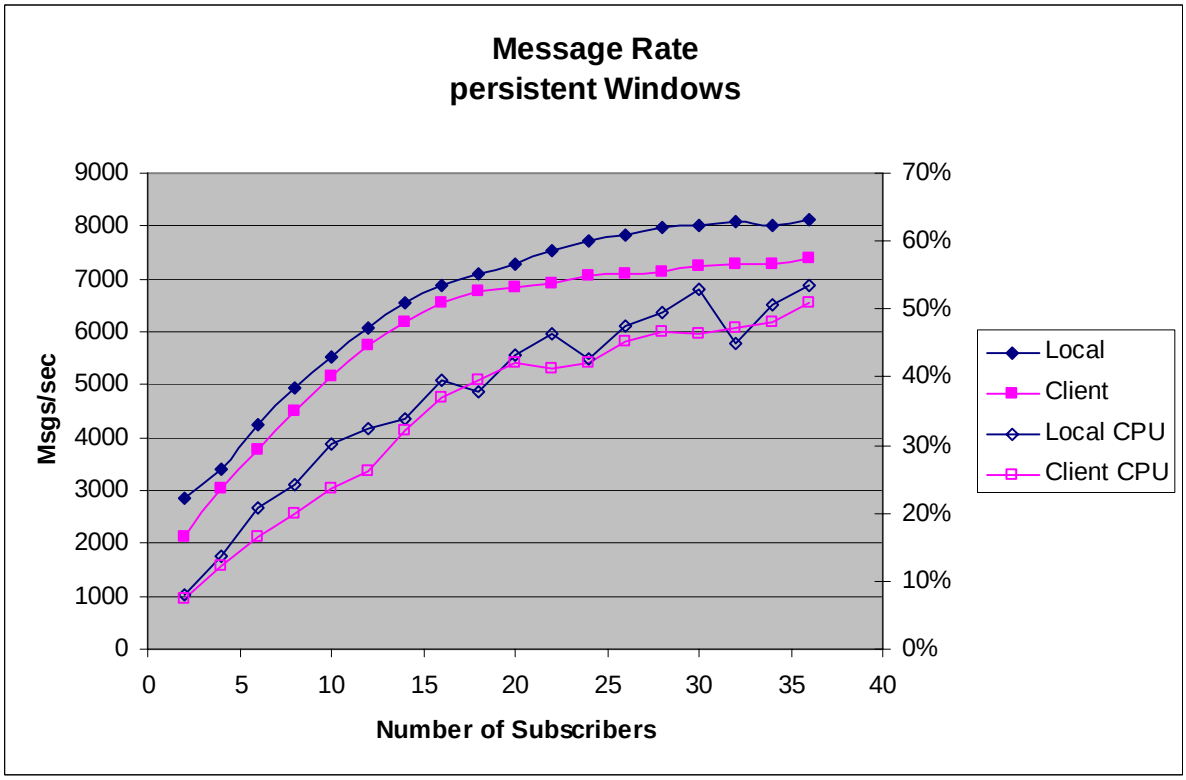


Chart 31 - Message rate persistent Windows

4.2 Maximum Publish/Subscribe message rate

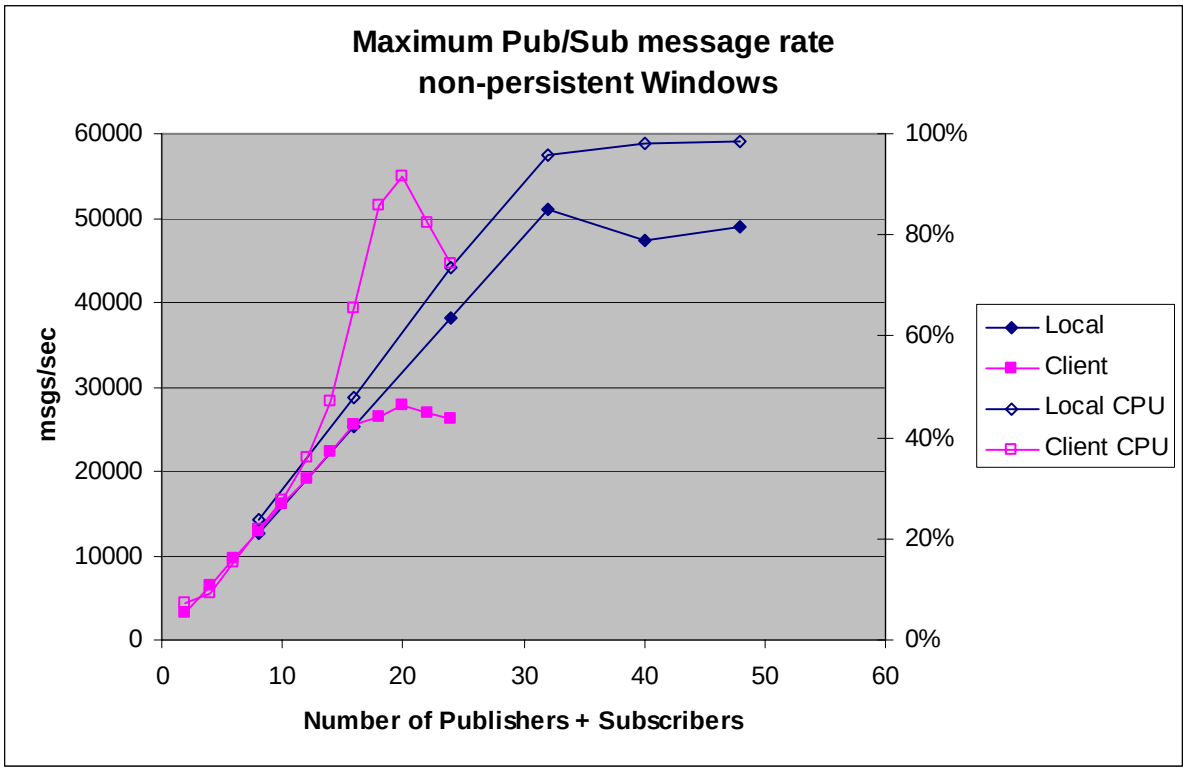


Chart 32 - Maximum Pub/Sub message rate non-persistent Windows

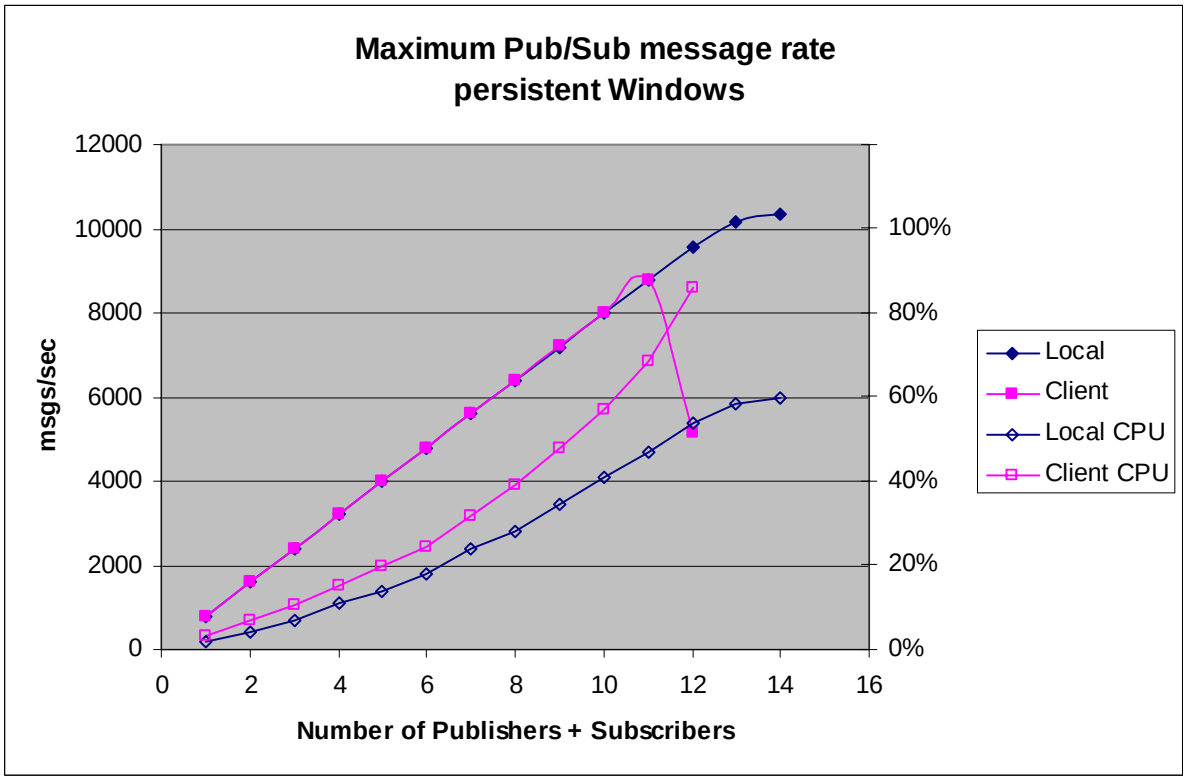


Chart 33 - Maximum Pub/Sub message rate persistent Windows

4.3 Comparison of Publish/Subscribe and Point-to-Point message rates

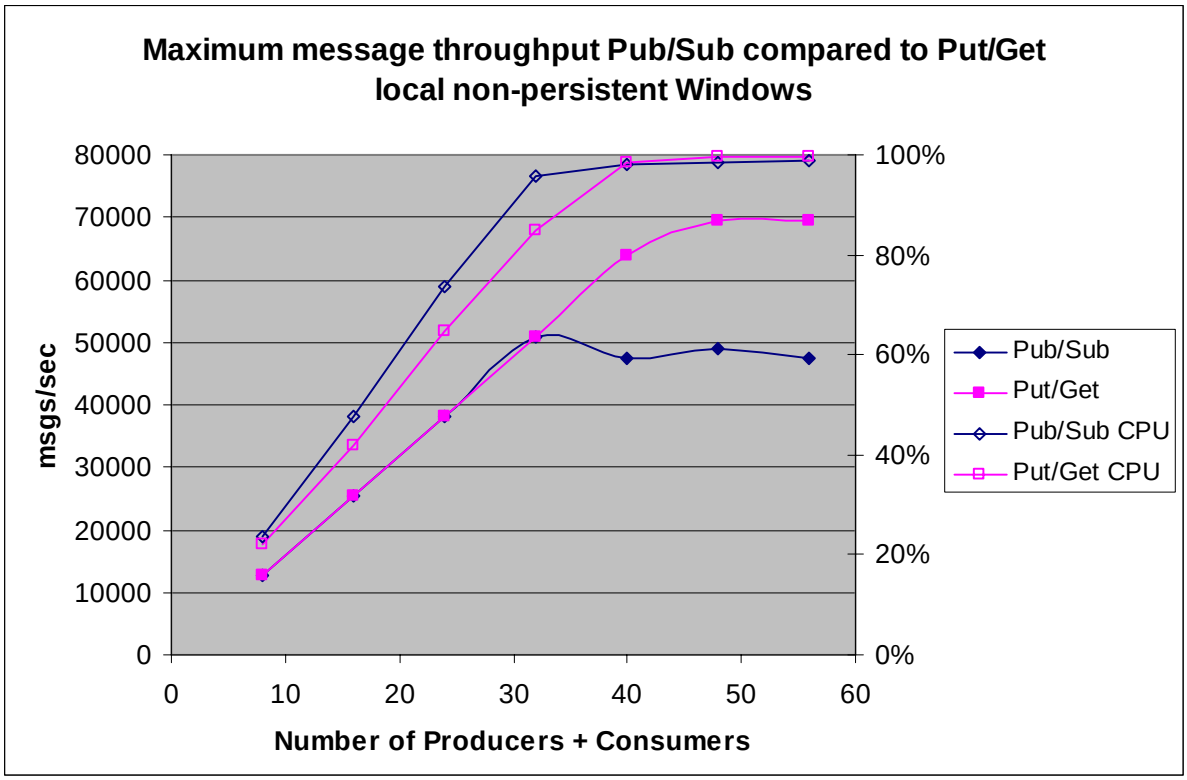


Chart 34 - Maximum message rate Pub/Sub compared to Put/Get local Windows

4.4 Comparison of MQI and JMS

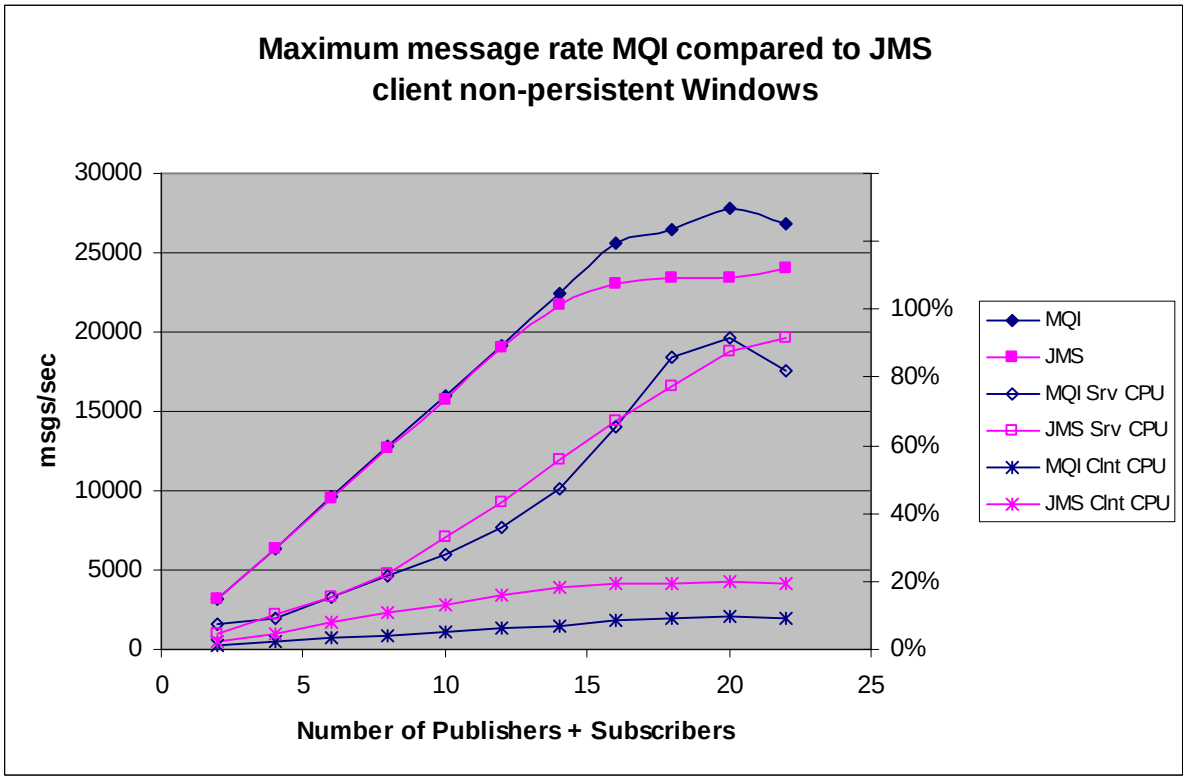


Chart 35 - Maximum message rate MQI compared to JMS client non-persistent Windows

4.5 Affect of message size on maximum message rate

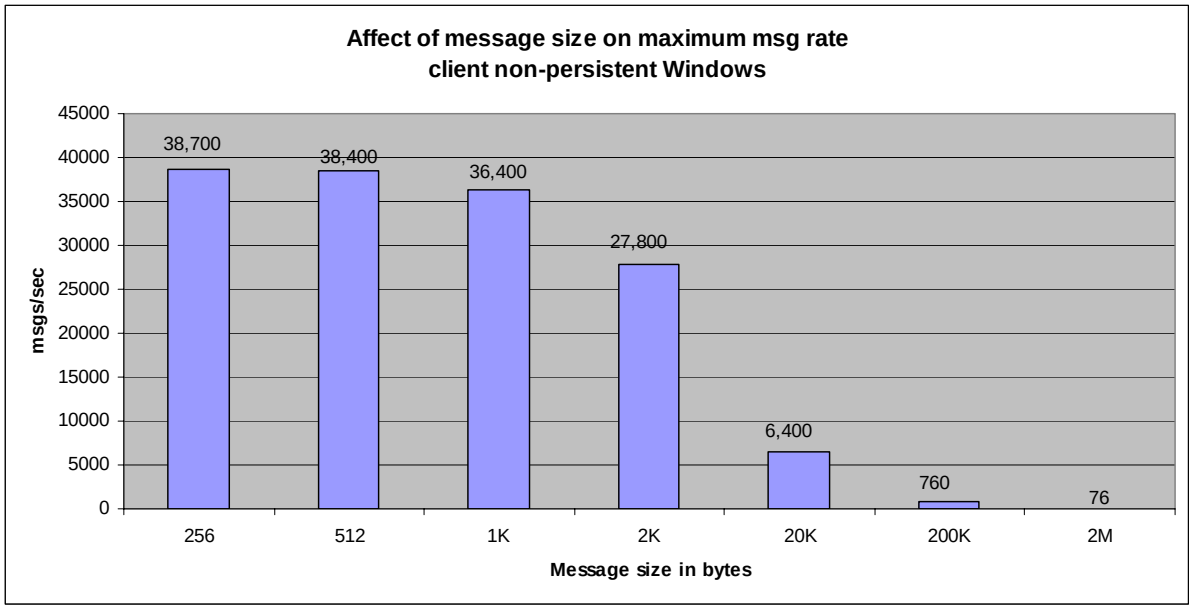


Chart 36 - Affect of message size on message rate client non-persistent Windows

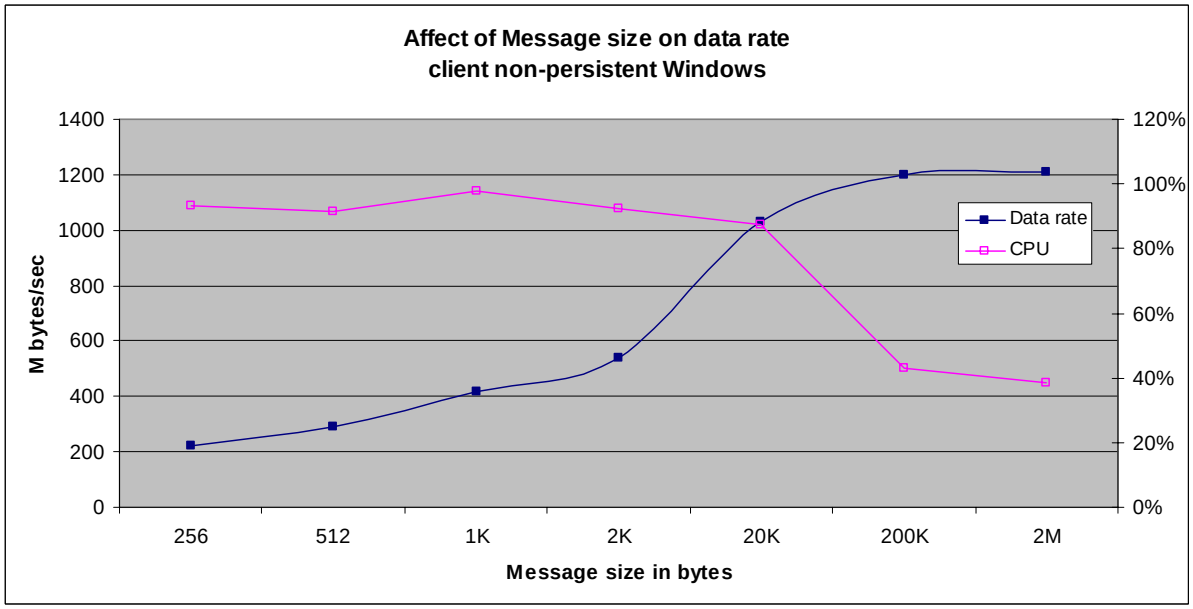


Chart 37 - Affect of message size on data rate client non-persistent Windows

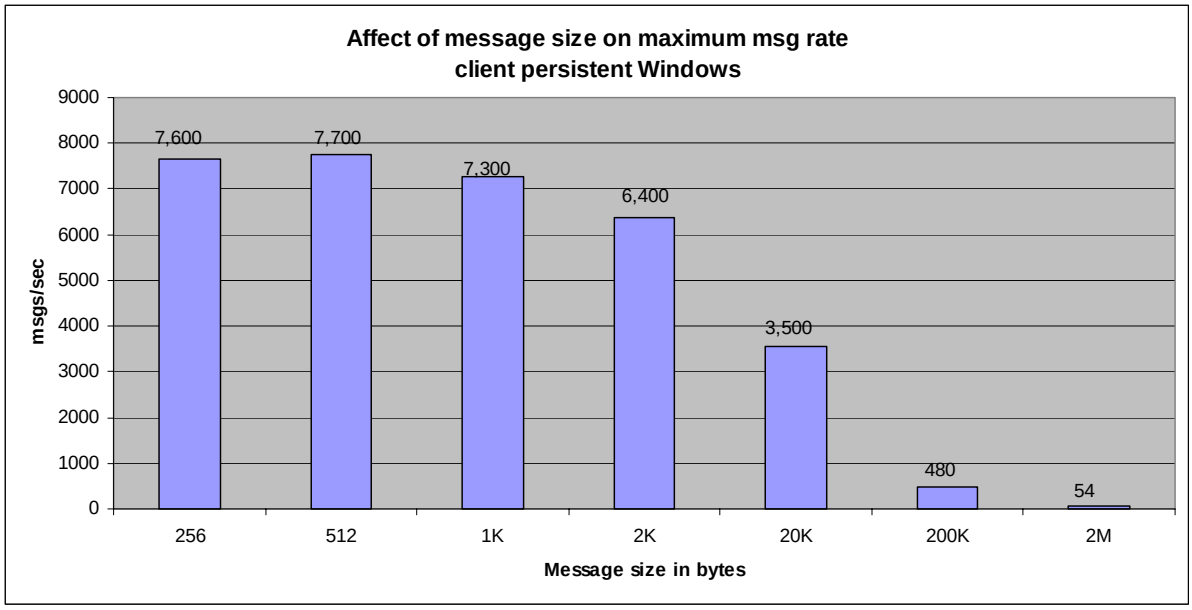


Chart 38 - Affect of message size on message rate client persistent Windows

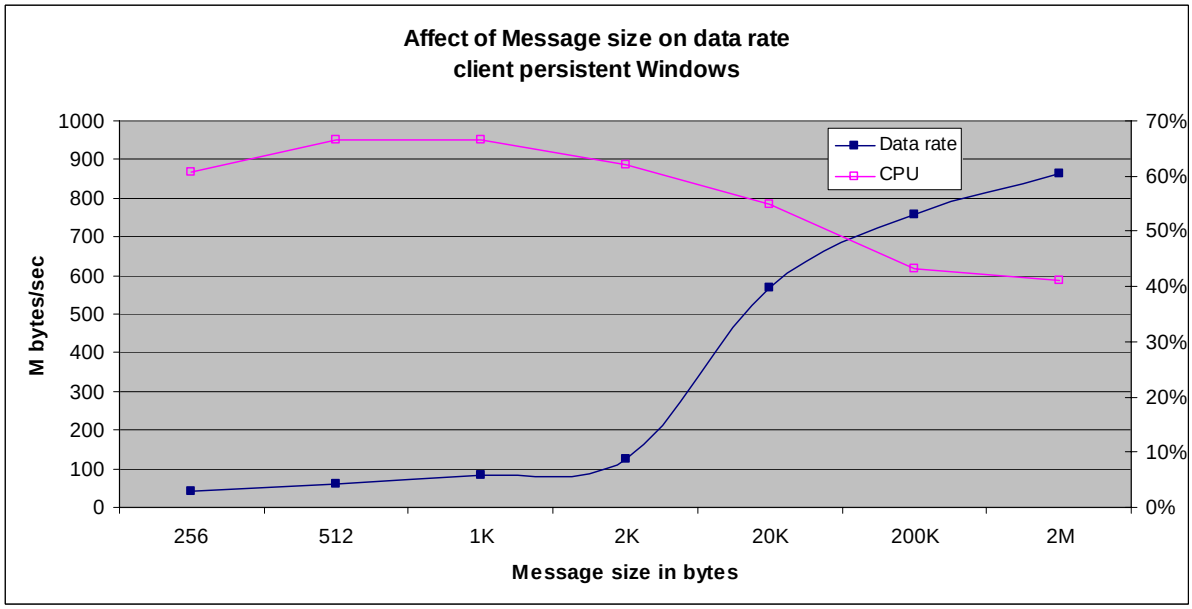


Chart 39 - Affect of message size on data rate client persistent Windows

5 Machine and Test Configurations

Clients are hosted by up to 4 Linux driver machines of varying powers. The driver machines are connected to the Windows machine over a 1Gb Ethernet LAN. The AIX and Linux64 machines are connected to the driver machines by a 10Gb LAN because they are more powerful machines and the additional network bandwidth is required to enable the servers to be driven to maximum CPU.

Due to the differences between the hardware used for each operating system, it is not possible to compare throughput across operating systems.

5.1 Linux64 Server

IBM System x3650 M3
2 x 6-core 2.80GHz Intel Xeon 5660 Hyper threading enabled
28 GB of RAM
Red Hat Enterprise Linux Server release 5.5 (kernel 2.6.18)
MQ Log and Queues on SAN disks on DS8700
10Gbit Ethernet Adapter

5.2 AIX Server

IBM Power7 P750 8233-E8B
16 core x 3.55GHz SMT x4 enabled
32 GB of RAM
AIX 7.1.0.0 TL05 SP2
MQ Log and Queues on SAN disks on DS8700
10Gbit Ethernet Adapter

5.3 Windows Server

IBM xSeries 350
4 core 2.80GHz Intel Xeon Hyper threading enabled
3.4GB of RAM
Windows Server 2003
MQ Log and Queues on SAN disks on DS8700
1Gbit Ethernet Adapter

5.4 SAN disk subsystem

The machines under test are connected to a SAN via a dedicated SVC. The SVC provides a transparent buffer between the server and SAN that will smooth any fluctuations in the response of the SAN due to external workloads. The server machines are connected via a fibre channel trunk to an 8Gb Brocade DCX director. The speed of each server is dictated by the server's HBA (typically 2Gb). 5GB generic LUNs are provisioned via SVC. The SVC is a 2145-8G4 which connects to the DCX at 4Gb. The SAN storage is provided by an IBM DS8700 which is connected to the DCX at 4Gb.

6 Tuning

Performance reports with tuning information for WebSphere MQ v7.1 (which is equally applicable to WebSphere MQ v7.5) on all supported operating systems can be found on the IBM SupportPac webpage at the following URL: WebSphere MQ family Performance Reports

6.1 Tuning the queue manager

The following tuning was applied to the queue manager (see qm.ini file) for the tests in this report:

- Log / LogBufferPages = 4096 (size of memory used to build log I/O records)
- Log / LogFilePages = 16348 (size of Log disk file extent)
- Log / LogPrimaryFiles = 16 (number of disks extents in log cycle)
- Log / LogType=CIRCULAR
- LogWriteIntegrity=TripleWrite
- Channels / MQIBindType = FASTPATH
- Channels/SHARECNV =1 (see below)
- TuningParameters / DefaultQBufferSize = 1MB (use 1MB of main memory per Q to hold non
- persistent messages before spilling to the file system) . This was set to 40MB for the 2MB message tests.
- TuningParameters / DefaultPQBufferSize = 1MB (use 1MB of main memory per Q to hold
- persistent messages). This was set to 40MB for the 2MB message tests.

6.2 Shared Conversations

Clients producing or consuming a small number of messages per second can usefully share the TCP socket with other threads in the same process. For MQ v7.5 the default is for 10 applications to share a channel and hence a TCP socket. Benchmarks that produce or consume multiple hundreds of messages per second will bottleneck on the shared socket and should use a single socket per application by setting SHARECNV=1.

6.3 Queue Buffer Size

Increasing the DefaultQBufferSize and DefaultPQBufferSize to 1MB is not advised on systems where memory is constrained and there are a large number of subscriptions which use managed queues.