WebSphere®MQ Publish Subscribe V7.0.1 - Performance Evaluations

Version 1.0

March 2010

Peter Toghill ,

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, March 2010.

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

Notices

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

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

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

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

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

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

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

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

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

wanting to understand the performance characteristics of *WebSphere MQ Publish Subscribe V7.0.1*. The information is not intended as the specification of any programming interface that is provided by WebSphere. It is assumed that the reader is familiar with the concepts and operation of WebSphere MQ V7.

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

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

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

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

- IBM
- WebSphere

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 the MQI clients supplied with WebSphere MQ for Linux V7.0.1, AIX V7.0.1, and Solaris V7.0.1 and is intended to assist with programming and capacity planning.

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 limits of WebSphere MQ Publish Subscribe.

The reader should have a general awareness of the Publish Subscribe API, Linux, and/or AIX 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 headlines of WebSphere MQ V7.0.1 Publish Subscribe
- WebSphere MQ messaging comparisons between Solaris, Linux and AIX

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

1 Overview 1
2 Client Channels Test Scenario 2
2.1 Publish/Subscribe Single Publisher, Many Subscribers Scenario(1:N) 2
2.2 Publish/Subscribe Single Publisher, Many Subscribers, Topic Cluster 3
2.3 Publish Subscribe multiple (Publisher, Topic, Subscriber) scenario 4
2.4 Point to Point multiple (Producer, Queue, Consumer) scenario 5
3 AIX Measurements 6
3.1.1 Publish Subscribe 1:N, Non Persistent messages 6
3.1.2 Publish Subscribe 1:N, Persistent messages 7
3.1.3 Publish Subscribe 1:N, Non Persistent messages, Cluster 8
3.1.4 Publish Subscribe 1:N, Persistent messages, Cluster 9
3.1.5 Publish Subscribe (Multiple P/T/S), Non Persistent messages 10
3.1.6 Publish Subscribe (Multiple P/T/S), Non Persistent messages 11
3.1.7 Publish Subscribe (Multiple P/T/S), Persistent messages 12
3.1.8 Point to Point (Multiple P/Q/C), Non Persistent messages 13
3.1.9 Point to Point (Multiple P/Q/C), Persistent messages 14
4 Linux Measurements 15
4.1.1 Publish Subscribe 1:N, Non Persistent messages 15
4.1.2 Publish Subscribe 1:N, Non Persistent messages, Cores 16
4.1.3 Publish Subscribe 1:N, Persistent messages 17
4.1.4 Publish Subscribe 1:N, Non Persistent messages, Cluster-4 18
4.1.5 Publish Subscribe 1:N, Non Persistent messages, Cluster Publications 19
4.1.6 Publish Subscribe 1:N, Persistent messages, Cluster 20
4.1.7 Publish Subscribe (Multiple P/T/S), Non Persistent messages 21
4.1.8 Publish Subscribe (Multiple P/T/S), Persistent messages 22
4.1.9 Point to Point (Multiple P/Q/C), Non Persistent messages 23
4.1.10 Point to Point (Multiple P/Q/C), Persistent messages 24
5 Solaris Measurements 25
5.1.1 Publish Subscribe 1:N, Non Persistent messages 25
5.1.2 Publish Subscribe 1:N, Persistent messages 26
5.1.3 Publish Subscribe 1:N, Non Persistent messages, Cluster 27
5.1.4 Publish Subscribe 1:N, Persistent messages, Cluster 28
5.1.5 Publish Subscribe (Multiple P/T/S), Non Persistent messages 29
5.1.6 Publish Subscribe (Multiple P/T/S), Persistent messages 30
5.1.7 Point to Point (Multiple P/Q/C), Non Persistent messages 31
5.1.8 Point to Point (Multiple P/Q/C), Persistent messages 32
6 Machine and Test Configurations 33
6.1 Linux, AIX, & Solaris Servers 33
6.2 SAN disk subsystem 34
6.3 Test case names 34
7 Summary 35
7.1 Publish Subscribe 1:N 35
7.2 Publish Subscribe (Multiple P/T/S) 35
8 Tuning 35

TABLES

Table 1 – Publish/Subscribe 1:N, non Persistent messages 6
Table 2 – Publish/Subscribe 1:N, Persistent messages 7
Table 3 – Publish/Subscribe 1:N, non Persistent messages, Cluster 8
Table 4 – Publish/Subscribe 1:N, Persistent messages, Cluster 9
Table 5 – Publish/Subscribe Multiple, non Persistent messages 10
Table 6 – Publish/Subscribe Multiple, non Persistent messages 11
Table 7 – Publish/Subscribe Multiple, Persistent messages 12
Table 8 – Point to point ,Multiple, non Persistent messages 13
Table 9 – Point to point ,Multiple, Persistent messages 14
Table 10 – Publish/Subscribe 1:N, non Persistent messages 15
Table 11 – Publish/Subscribe 1:N, non Persistent messages 16
Table 12 – Publish/Subscribe 1:N, Persistent messages 17
Table 13 – Publish/Subscribe 1:N, non Persistent messages, Cluster 18
Table 14 – Publish/Subscribe 1:N, non Persistent messages, Cluster 19
Table 15 – Publish/Subscribe 1:N, Persistent messages, Cluster 20
Table 16 – Publish/Subscribe Multiple, non Persistent messages 21
Table 17 – Publish/Subscribe Multiple, Persistent messages 22
Table 18 – Point to point ,Multiple, non Persistent messages 23
Table 19 – Point to point ,Multiple, Persistent messages 24
Table 20 – Publish/Subscribe 1:N, non Persistent messages 25
Table 21 – Publish/Subscribe 1:N, Persistent messages 26
Table 22 – Publish/Subscribe 1:N, non Persistent messages, Cluster 27
Table 23 – Publish/Subscribe 1:N, Persistent messages, Cluster 28
Table 24 – Publish/Subscribe Multiple, non Persistent messages 29
Table 25 – Publish/Subscribe Multiple, Persistent messages 30
Table 26 – Point to point ,Multiple, non Persistent messages 31
Table 27 – Point to point ,Multiple, Persistent messages 32

FIGURES

Figure 1 – MQI-client channels into a remote queue manager 2
Figure 2 – Publish Subscribe 1:N 2
Figure 4 – Publish Subscribe 4
Figure 5 Multiple (Producer Queue Consumer) 5
Figure 6 – Publish Subscribe 1:N, non persistent 6
Figure 7 – Publish Subscribe 1:N, persistent 7
Figure 8 – Publish Subscribe 1:N, non persistent, Cluster 8
Figure 9 – Publish Subscribe 1:N, persistent, cluster 9
Figure 10 – Publish Subscribe Multiple, non persistent 10
Figure 11 – Publish Subscribe Multiple, non persistent 11
Figure 12 – Publish Subscribe multiple, persistent 12
Figure 13 – Point to Point, Multiple, non persistent 13
Figure 14 – Point to point, multiple, persistent 14
Figure 15 – Publish Subscribe 1:N, non persistent 15
Figure 16 – Publish Subscribe 1:N, non persistent, Cores 16
Figure 17 – Publish Subscribe 1:N, persistent 17
Figure 18 – Publish Subscribe 1:N, non persistent, Cluster 18
Figure 19 – Publish Subscribe 1:N, non persistent, Cluster publications 19
Figure 20 – Publish Subscribe 1:N, persistent, cluster 20
Figure 21 – Publish Subscribe Multiple, non persistent 21
Figure 22 – Publish Subscribe Multiple, persistent 22
Figure 23 – Point to Point, Multiple, non persistent 23
Figure 24 – Point to point, multiple, persistent 24
Figure 25 – Publish Subscribe 1:N, non persistent 25
Figure 26 – Publish Subscribe 1:N, persistent 26
Figure 27 – Publish Subscribe 1:N, non persistent, Cluster 27
Figure 28 – Publish Subscribe 1:N, persistent, cluster 28
Figure 29 – Publish Subscribe Multiple, non persistent 29
Figure 30 – Publish Subscribe Multiple, persistent 30
Figure 31 – Point to Point, Multiple, non persistent 31
Figure 32 – Point to point, multiple, persistent 32

1 Overview

The two Publish_Subscribe and one Point to Point scenarios used in Chapter 3, 4, & 5 in this report are measured and reported with Persistent and non Persistent messages on Linux, AIX, and Solaris systems.

1) Publish Subscribe (single publisher, single topic, multiple subscribers)
2) Multiple sets of Publisher Topic Subscriber (single publisher, single topic, single subscribers)
3) Multiple sets of Producer Queue Consumer (single producer, single queue, Single consumers)

- The message format used is a 2048 byte character message.
- In addition there are some measurements of clustered Publish-Subscribe engines
- Persistent messages are transactional. (MQCmit issued by application for each message). This also significantly improves throughput when multiple threads are processing messages on the same queue especially when using non cached disks for the MQ Log.
- The 'multiple sets' message producers insert messages at a fixed rate of 1600 non persistent per second or 400 persistent messages per second.
- Publisher and Subscriber Client applications are written on MQI 'C' language
- Messages Producer/Consumer are located on Linux driver systems for these 'Client' measurements.
- Each sample point reported is the average of two minutes of data collection.
- Clients run on Linux and the bottleneck with these Client measurements is the server because adequate power is available in the driving machines.

2 Client Channels Test Scenario

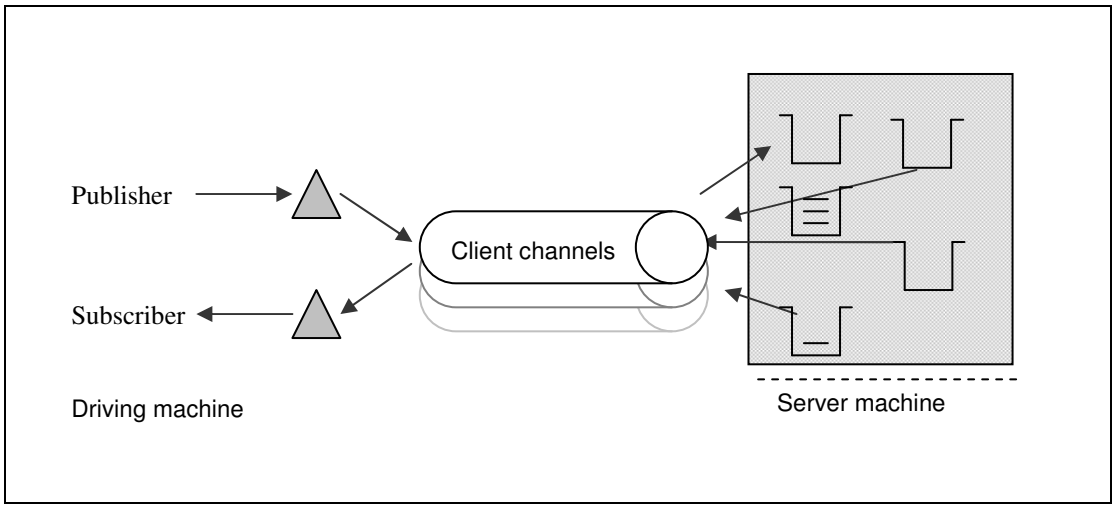

Figure 1 – MQI-client channels into a remote queue manager

The various message producers publish a message (over a client channel), to the relevant topic on the server. The subscriber application waits indefinitely for messages to arrive on its input queue. In the Point to Point scenario, the Producer and Consumer replace the Publisher and Subscriber. The Producrs and Consumers are spread over several different driver machines.

The Client Channel is set to 'MQIBindType = FASTPATH' . The major benefit is for non persistent messages because it eliminates the AGENT process (AMQZLAA) and reduces CPU cost. Environments using Channel exits should be aware that the exit code would run inside the Queue Manager.

2.1 Publish/Subscribe Single Publisher, Many Subscribers Scenario(1:N)

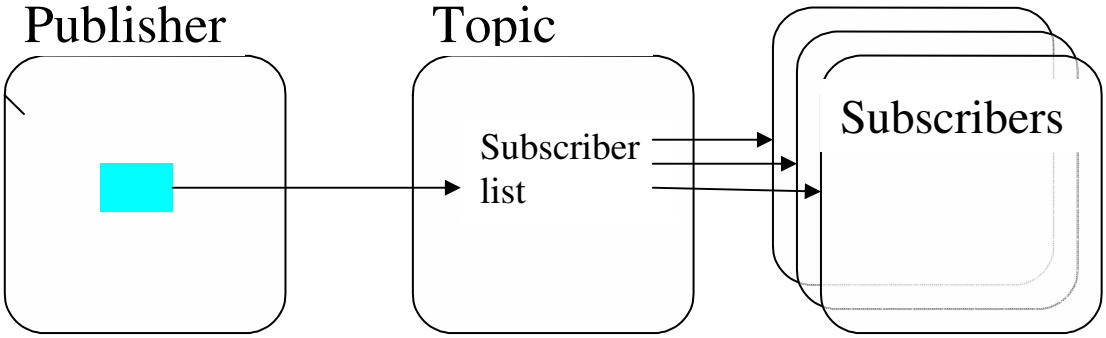

Figure 2 – Publish Subscribe 1:N

All subscribers used unique subscriber queues. Persistent subscribers received five messages in each transaction.

1 A publisher publishes a message to the single topic.
2 Each subscriber then receives the message.

This testcase provides asynchronous messaging since there is no connection between the number of messages in the system and the number of publishers or subscribers. The publisher publishes the next message without any 'think' time. Message count is the number of published messages plus those consumed by the subscribers.

2.2 Publish/Subscribe Single Publisher, Many Subscribers, Topic Cluster

Clustered topics enable subscribers to be spread over multiple Queue managers. Each Queue manager in the cluster will be notified of publications on the topic if they have relevant subscribers so they can create messages for each subscriber. Subscribers are evenly distributed between the clustered Queue Managers.

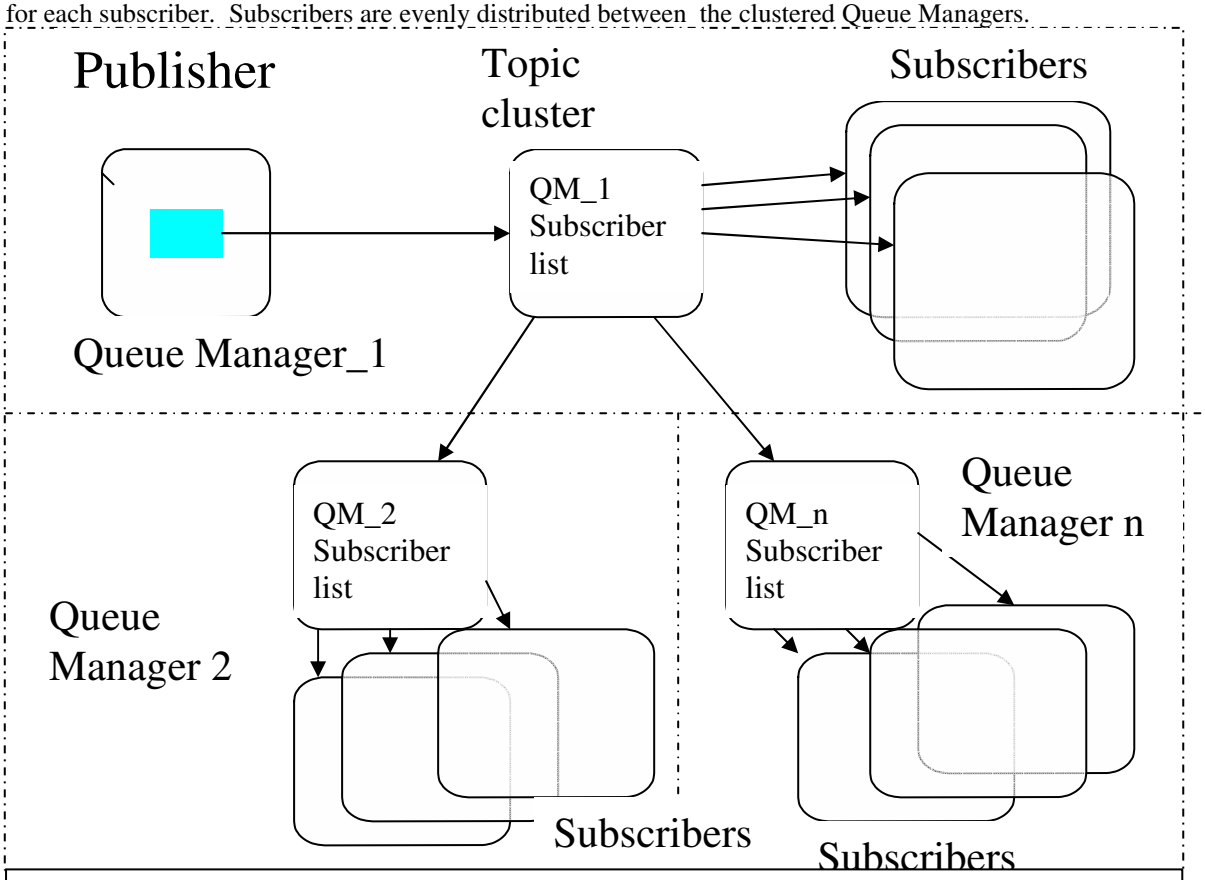

Figure 3 Pub Sub Clusters

2.3 Publish Subscribe multiple (Publisher, Topic, Subscriber) scenario

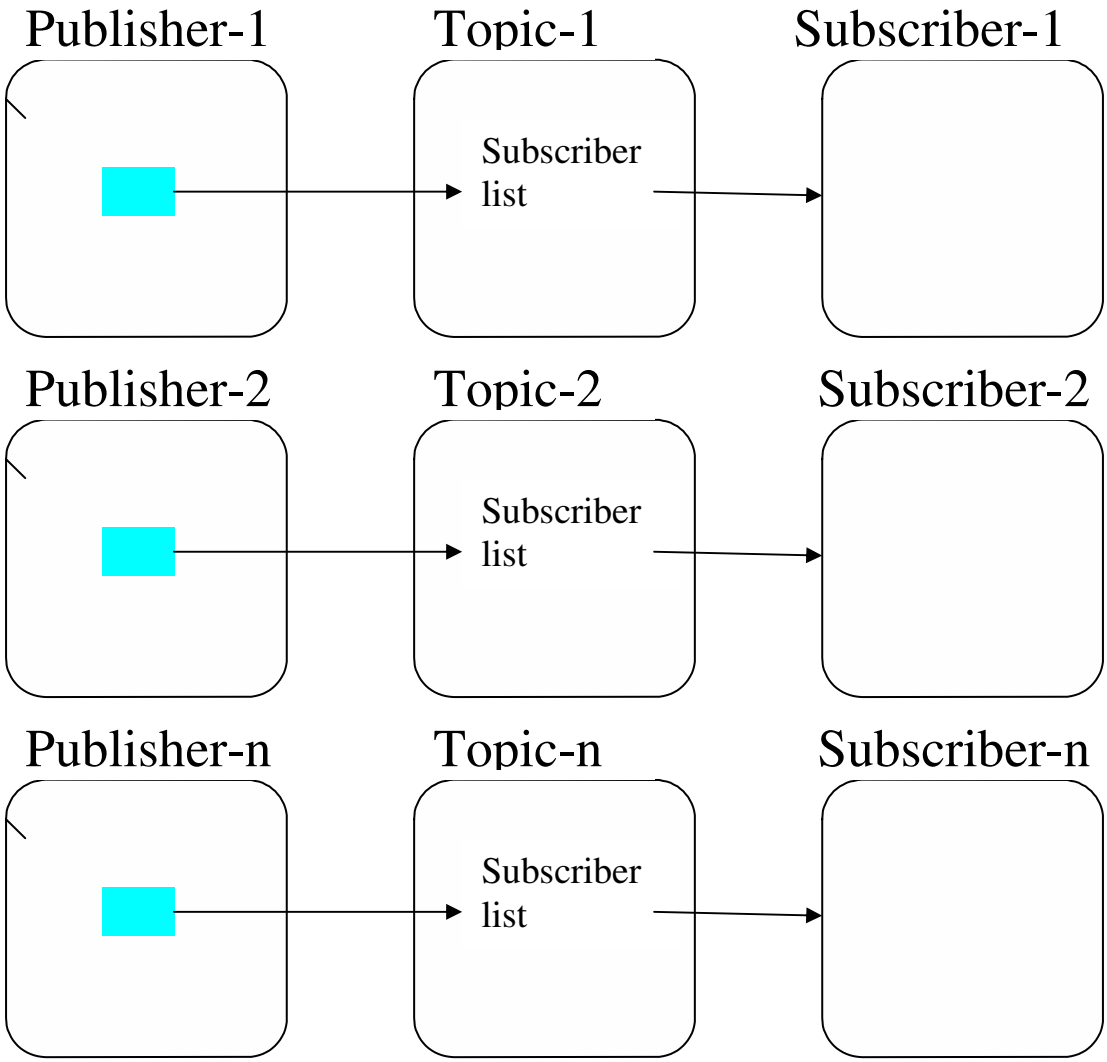

Figure 4 – Publish Subscribe

All subscribers used unique subscriber queues. Persistent subscribers received five messages in each transaction.

1 A publisher publishes a message to the single topic.

2 Only one subscriber had registered for the topic then receives the message.

This testcase provides asynchronous messaging since there is no connection between the number of messages in the system and the number of publishers or subscribers. The publisher publishes message at a predetermined rate which provides for a gradually increasing workload as the number of (Publisher, Topic, Subscriber) triplets is increased. The message production rate per publisher is 1600 per second for non-persistent and 400 per second for persistent messages. Message count is the number of published messages plus those consumed by the subscribers.

2.4 Point to Point multiple (Producer, Queue, Consumer) scenario

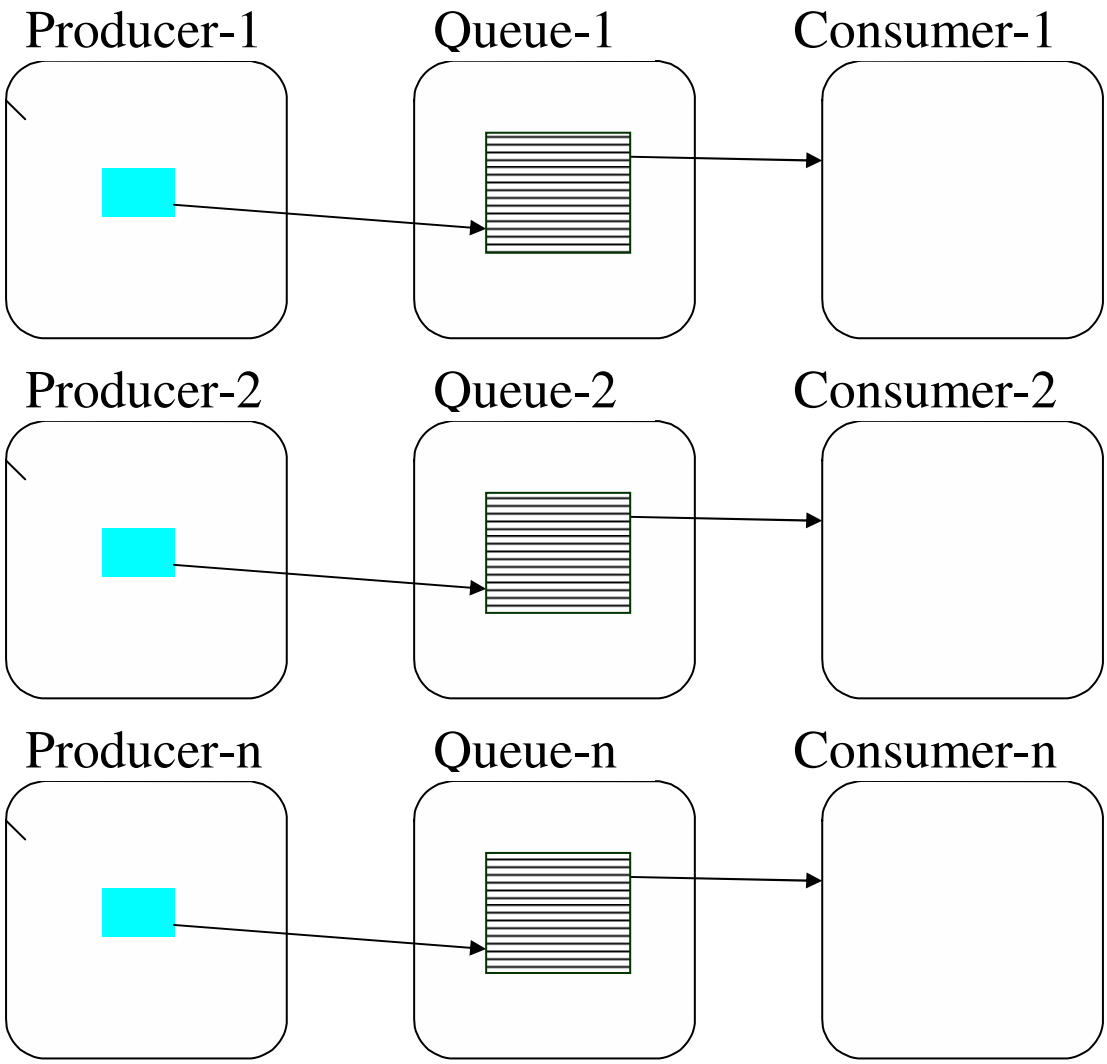

Figure 5 Multiple (Producer Queue Consumer)

This testcase provides asynchronous messaging since there is no connection between the number of messages in the system and the number of message producers. Messages are produced at a predetermined rate by each producers (1600 per second for non-persistent or 400 per second for persistent messages.) This provides for a gradually increasing workload as the number of (Producers, Queues, Consumer) triplets is increased. Message count is the number of messages produced plus the number consumed.

3 AIX Measurements

3.1.1 Publish Subscribe 1:N, Non Persistent messages

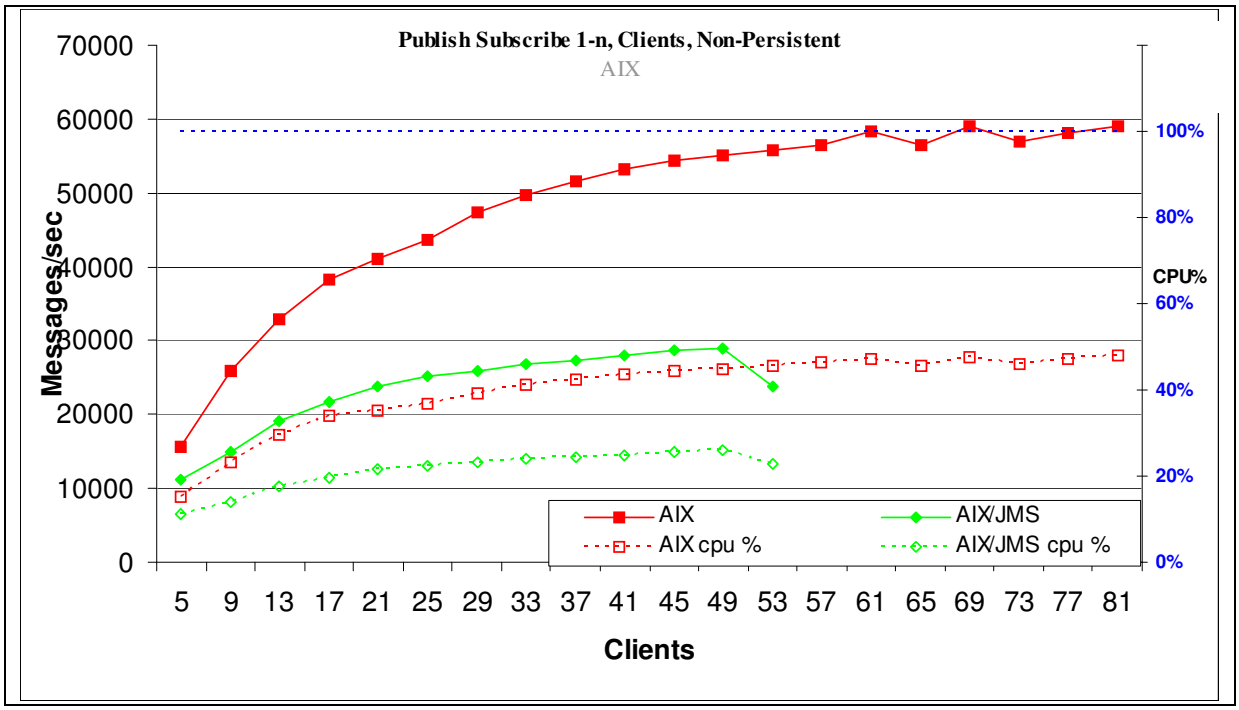

Figure 6 – Publish Subscribe 1:N, non persistent

Test name: **APSN**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 4 publication**
MQI	**81**	**59095**	**730**	**48%**	**3128**
JMS	**49**	**28922**	**590**	**26%**	**2258**

Table 1 – Publish/Subscribe 1:N, non Persistent messages

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 3128 publications per second can be achieved using MQI. The response time for the publish command increases as the number of subscribers increase. With 80 subscribers, the publisher creates 730 messages per second which are all consumed by the subscribers. The server cost to process a JMS message is about 6% more than processing a message from an MQI/C program. This is due to the longer datastream which includes JMS properties.

The JAVA/JMS code in the client uses significantly more CPU in the client/driver machine compared with the MQI/C client. This slows down the message production rate and hence the work submitted to the server is only half that achieved by the MQI client. All subsequent measurements in this report use the MQI client with 'C' language interface. MQ/JMS Pub/Sub measurements are contained in MP07.

3.1.2 Publish Subscribe 1:N, Persistent messages

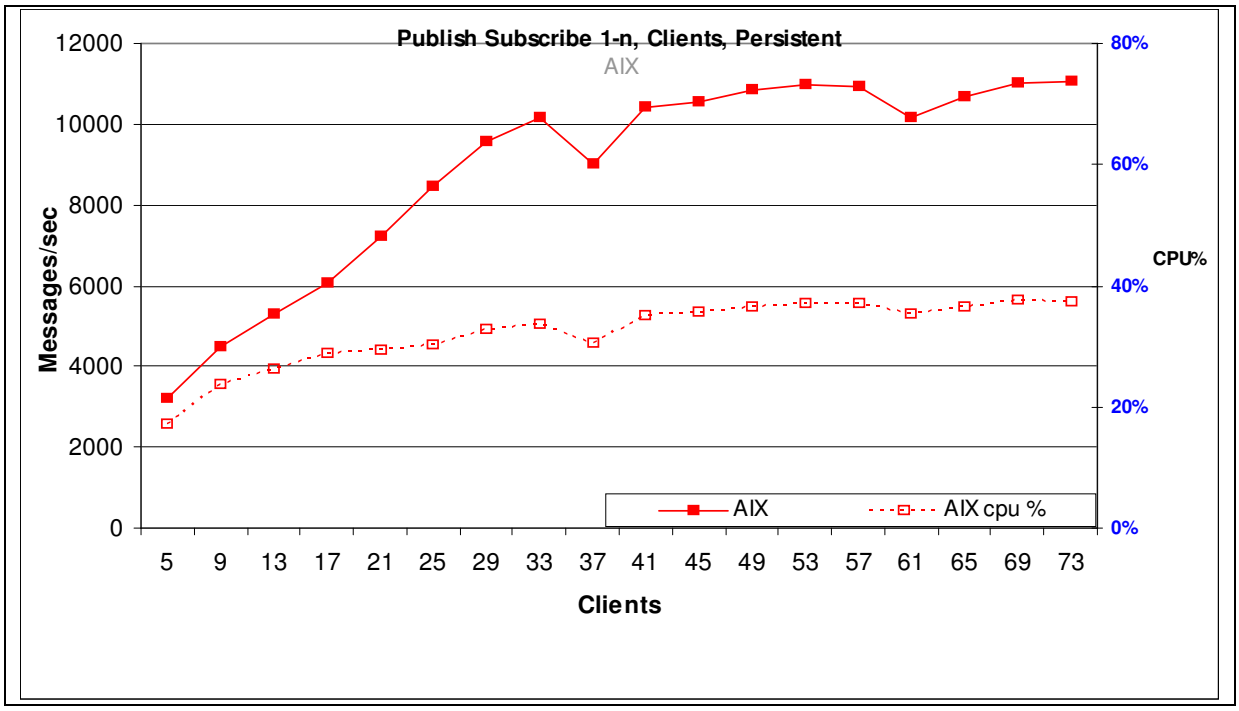

Figure 7 – Publish Subscribe 1:N, persistent

Test name: **APSP**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 4 subscribers Per publication**
Aix/MQI	**73**	**11047**	**151**	**37%**	**640**

Table 2 – Publish/Subscribe 1:N, Persistent messages

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 640 publications per second can be achieved on AIX. The response time for the publish command increases as the number of subscribers increase. On AIX with 72 subscribers, the publisher creates 151 messages per second which are all consumed by the subscribers

3.1.3 Publish Subscribe 1:N, Non Persistent messages, Cluster

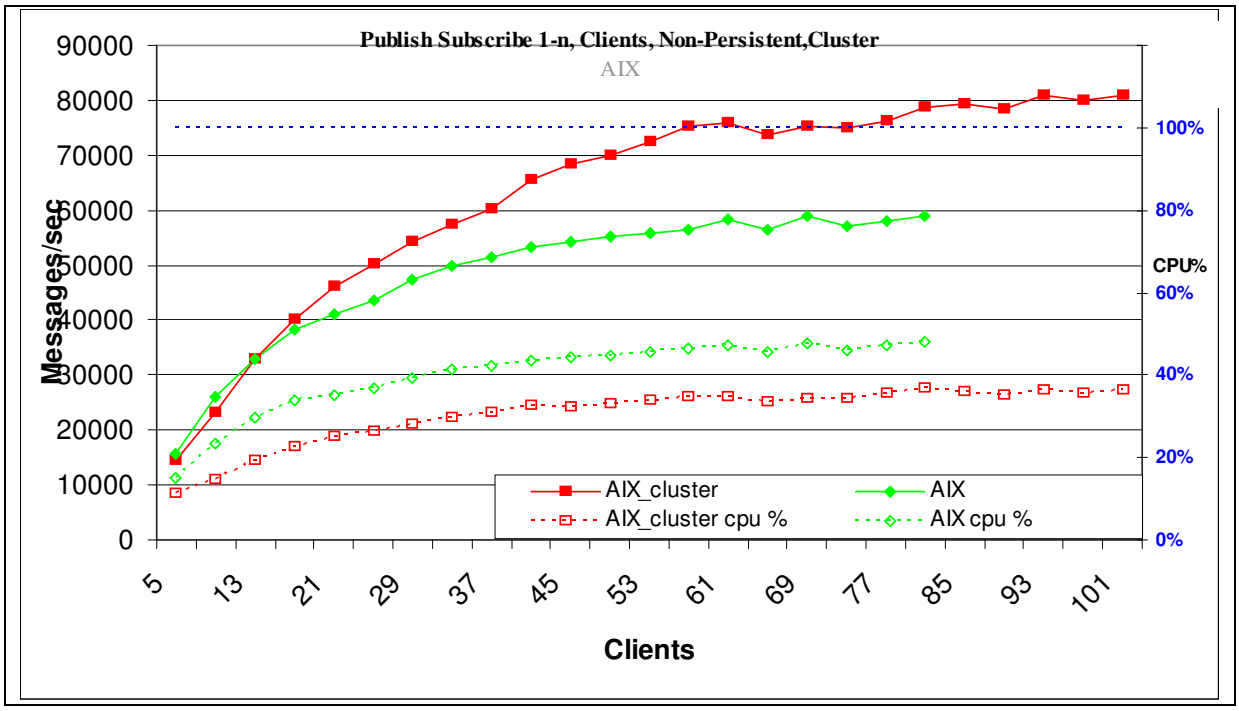

Figure 8 – Publish Subscribe 1:N, non persistent, Cluster

Test name: **APSN_C**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second With 4 subscribers Per publication**
				P6	**P5**	
MQI	**81**	**59095**	**730**	**48%**		**3128**
MQI/Cluster	**81** **101**	**78733** **81059**	**972** **802**	**36%** **36%**	**53%** **53%**	**2888**

Table 3 – Publish/Subscribe 1:N, non Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 3128 publications per second can be achieved by a single pub_sub engine and 2888 by a cluster of 2 engines when the clients are split between the pub-sub engines. With more than 12 clients higher throughput can be obtained from a cluster . With under 12 subscribers to a topic, the overhead of supporting clustering means that a non clustered system provide better throughput. With a small number of subscribers , the degradation can be 30%.

The additional machine in the cluster has increased the maximum publication rate for 80 subscribers by 33% from 730 to 972 per second. The second machine (Power5) was less powerful than the original machine (Power 6). An identical machine in the cluster would be expected to provide an increased in thoughput of between 60% and 70% because the second QM has more work to do than the original QM. Half of the subscribers are attached to each Queue Manager. The publisher is attached to the original machine (P6) but the second machine has more work to do in accepting the publications from P6 and fanning them out to its subscribers.

3.1.4 Publish Subscribe 1:N, Persistent messages, Cluster

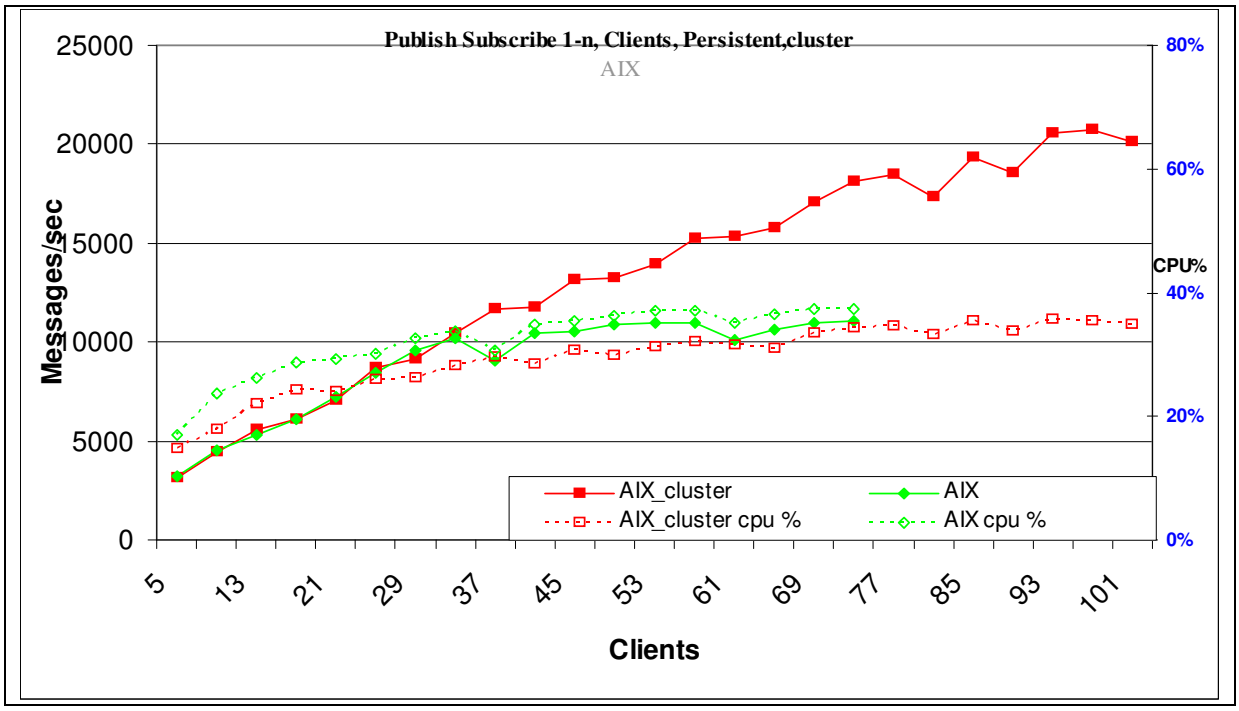

Figure 9 – Publish Subscribe 1:N, persistent, cluster

Test name: **APSP_C**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second With 2 subscribers Per publication**
				P6	**P5**	
AIX/MQI	**73**	**11047**	**151**	**37%**		**640**
MQI/Cluster	**97**	**20745**	**213**	**35%**	**49%**	**640**

Table 4 – Publish/Subscribe 1:N, Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 640 publications per second can be achieved on both the single and clustered pub-sub engine. The clustered system has two queue manager to handle the increased persistent messaging load and this enables it to provide an increased capacity of 88%. (Note that twice as much server hardware is being used). The second machine (P5) is less powerful and has more work to do than the original machine(P6) where the publisher is attached.

3.1.5 Publish Subscribe (Multiple P/T/S), Non Persistent messages

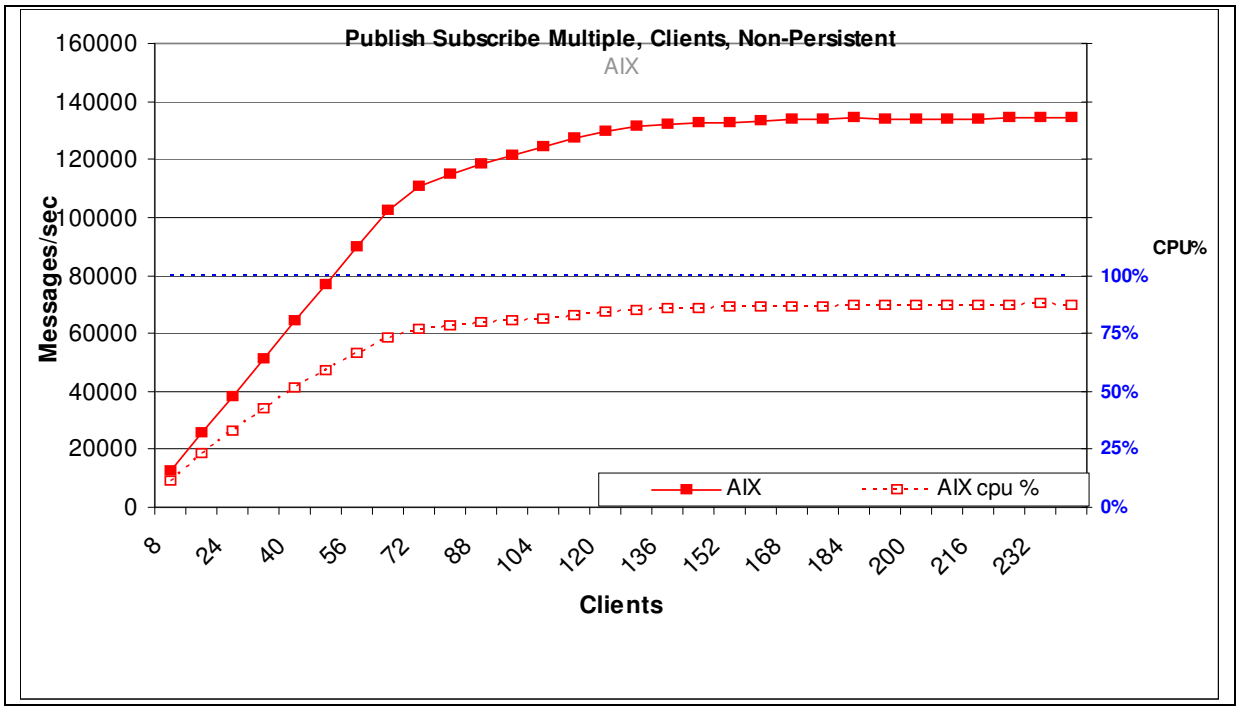

Figure 10 – Publish Subscribe Multiple, non persistent

Test name: **APTN**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
AIX/MQI	**64**	**102295**	**1598**	**73%**
	184	**134246**	**730**	**87%**

Table 5 – Publish/Subscribe Multiple, non Persistent messages

There are two clients in each (Publisher/Topic/Subscriber) group. Each publisher creates 1600 non persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 32 producers and 32 consumers , the expected throughput is 1600*32*2=102400 whereas the measured throughput is 102295 messages per second. Adding additional publishers causes the cpu to exceeed 75% busy which causes the publication rate per publisher to slow down until the system reaches its maximum throughput of 134246 messages with 184 publishers

3.1.6 Publish Subscribe (Multiple P/T/S), Non Persistent messages

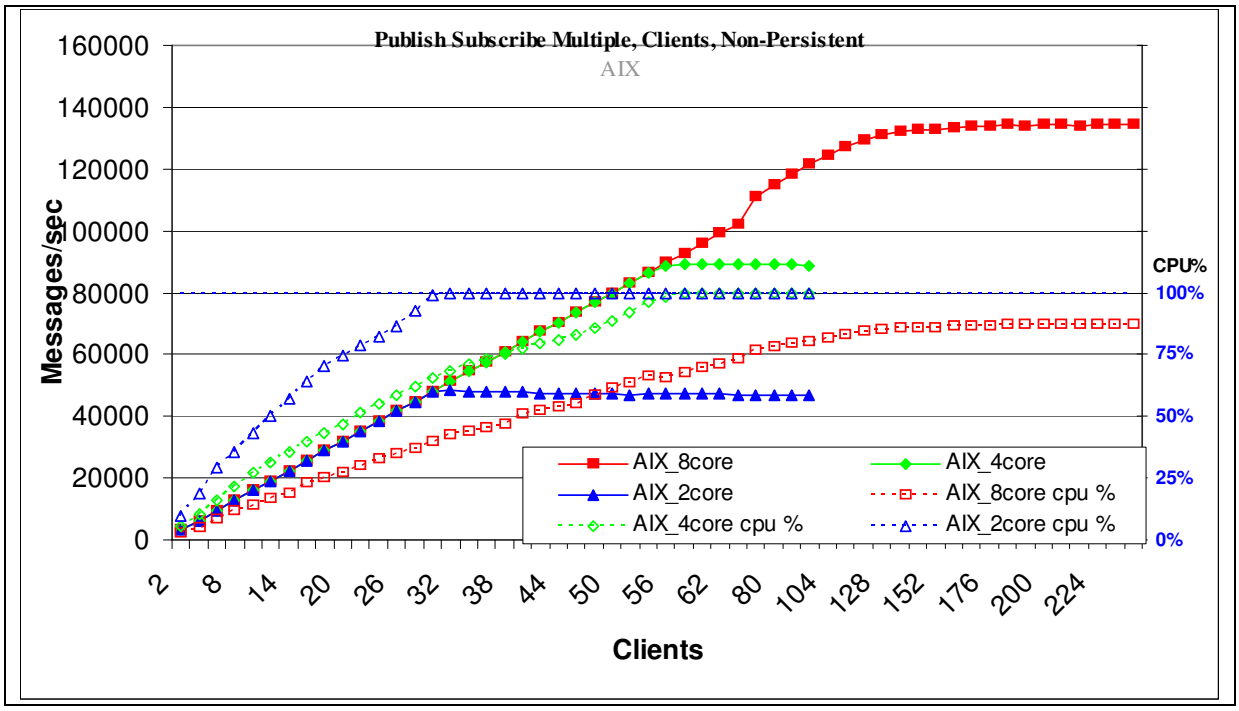

Figure 11 – Publish Subscribe Multiple, non persistent

Test name: **APTN**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
AIX_2core	**34**	**48088**	**1414**	**100%**
AIX_4core	**54**	**86317**	**1598**	**96%**
AIX_8core	**72** **120**	**110815** **129403**	**1539** **1078**	**71%** **77%**

Table 6 – Publish/Subscribe Multiple, non Persistent messages

Each publisher creates 1600 non persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 2 core and 4 cores this happens when the CPU reaches 100%. With 8 cores the message rate per publisher is over 95% of the requested rate for 72 clients but additional Publishers cause the rate per publisher to degrade.

Doubling the number of cores from 2 to 4 increases the workload by 80%. Doubling from 4 to 8 cores increases throughput by 50%.

3.1.7 Publish Subscribe (Multiple P/T/S), Persistent messages

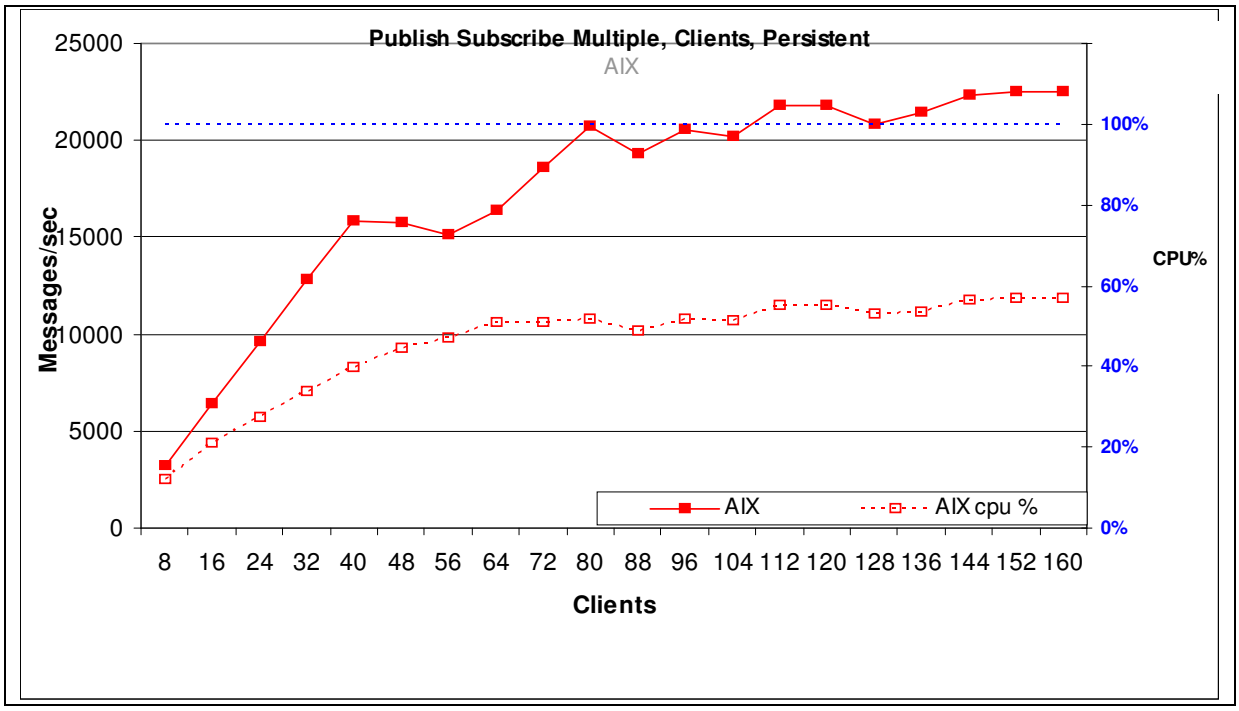

Figure 12 – Publish Subscribe multiple, persistent

Test name: **APTP**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
AIX/MQI	**40**	**15876**	**397**	**40%**
	80	**20750**	**259**	**51%**

Table 7 – Publish/Subscribe Multiple, Persistent messages

Each message producer creates 400 persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 20 producers and 20 consumers (40 Clients) on AIX, the expected throughput is 400*20*2=16000 whereas the measured throughput is 15876 messages per second. Additional Publishers cause a slowdown in the rate per publisher but the overall system messaging rate continues to gradually increase. 40 publishers (+ 40 subscribers) cause the effective publication rate to be reduced to 260 per second

3.1.8 Point to Point (Multiple P/Q/C), Non Persistent messages

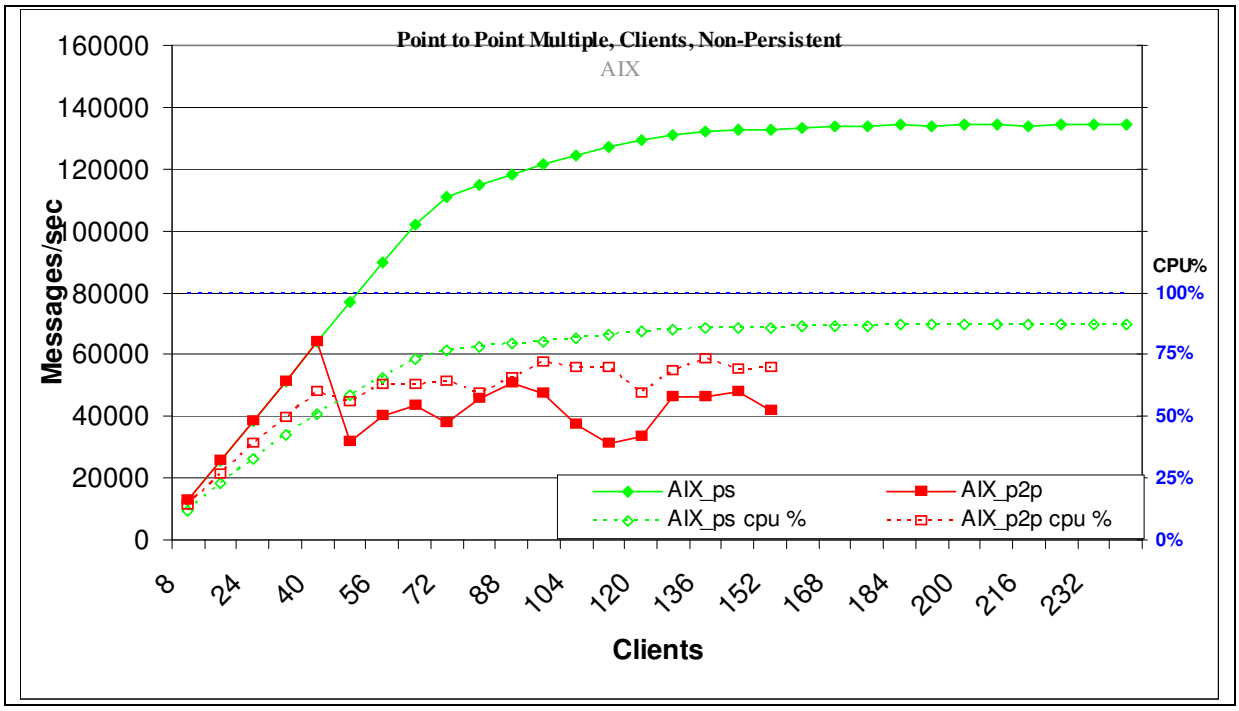

Figure 13 – Point to Point, Multiple, non persistent

Test name: **ATPN**	**Clients**	**Messages Per second**	**Server CPU**
Aix_PubSub	**64** **132**	**102295** **132578**	**73%** **86%**
Aix_P2P	**40**	**63851**	**60%**

Table 8 – Point to point ,Multiple, non Persistent messages

Each publisher or Producer creates 1600 non persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 32 Publishers and 32 subscribers (64 Clients) on AIX, the expected throughput is 1600*32*2=102400 whereas the measured throughput is 102295 messages per second. The Point to Point (P2P) scenario has 20 Producers and 20 consumers (40 clients) with an expected throughput rate of 1600*20*2 = 64000 and a measured throughput of 63851. Additional clients cause a reduction in system capacity.

3.1.9 Point to Point (Multiple P/Q/C), Persistent messages

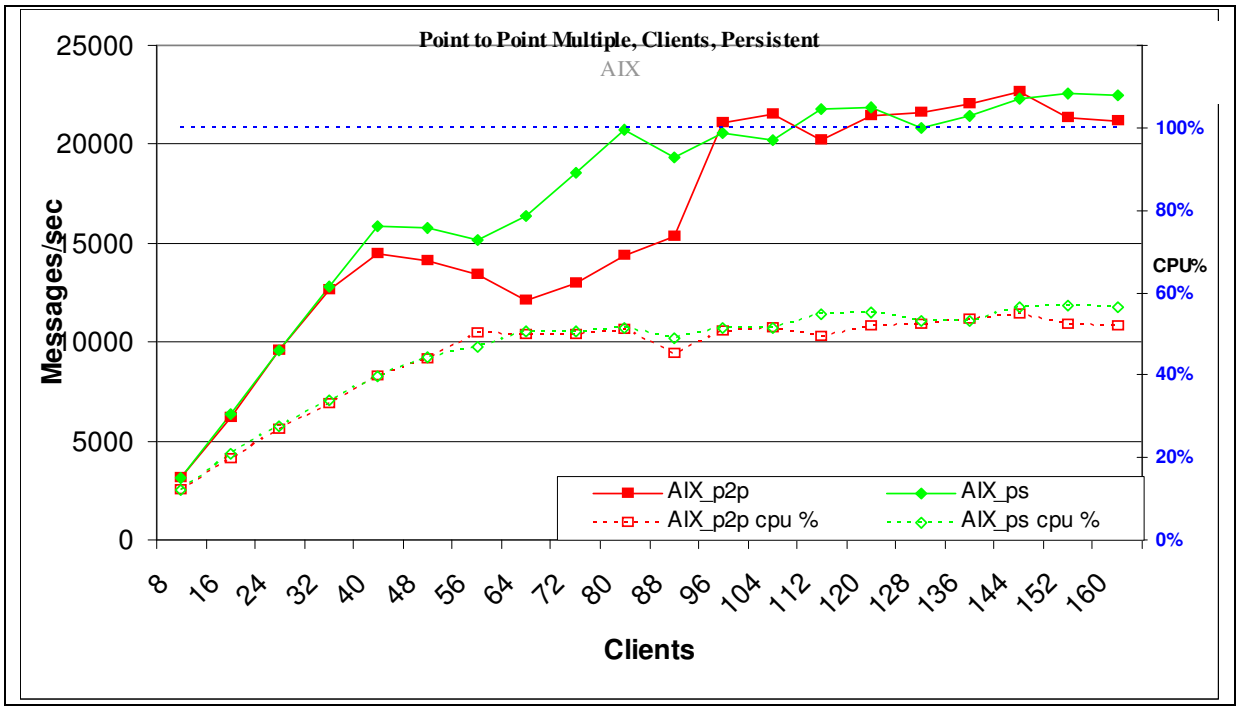

Figure 14 – Point to point, multiple, persistent

Test name: **ATPP**	**Clients**	**Messages Per second**	**Server CPU**
Aix_PubSub	**40** **96**	**15876** **20516**	**40%** **51%**
Aix-P2P	**40** **96**	**14420** **21047**	**40%** **51%**

Table 9 – Point to point ,Multiple, Persistent messages

Each message producer creates 400 messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 20 producers and 20 consumers (40 clients) on AIX, the expected throughput is 400*20*2=16000 whereas the measured Pub/Sub throughput is 15876 messages per second and 14420 point to point. Additional clients increase the message capacity to over 20000.

4 Linux Measurements

The majority of measurements were made with an x-Series x3850 system but some comparisons are made with the x-Series x366 system or x7350 system.

4.1.1 Publish Subscribe 1:N, Non Persistent messages

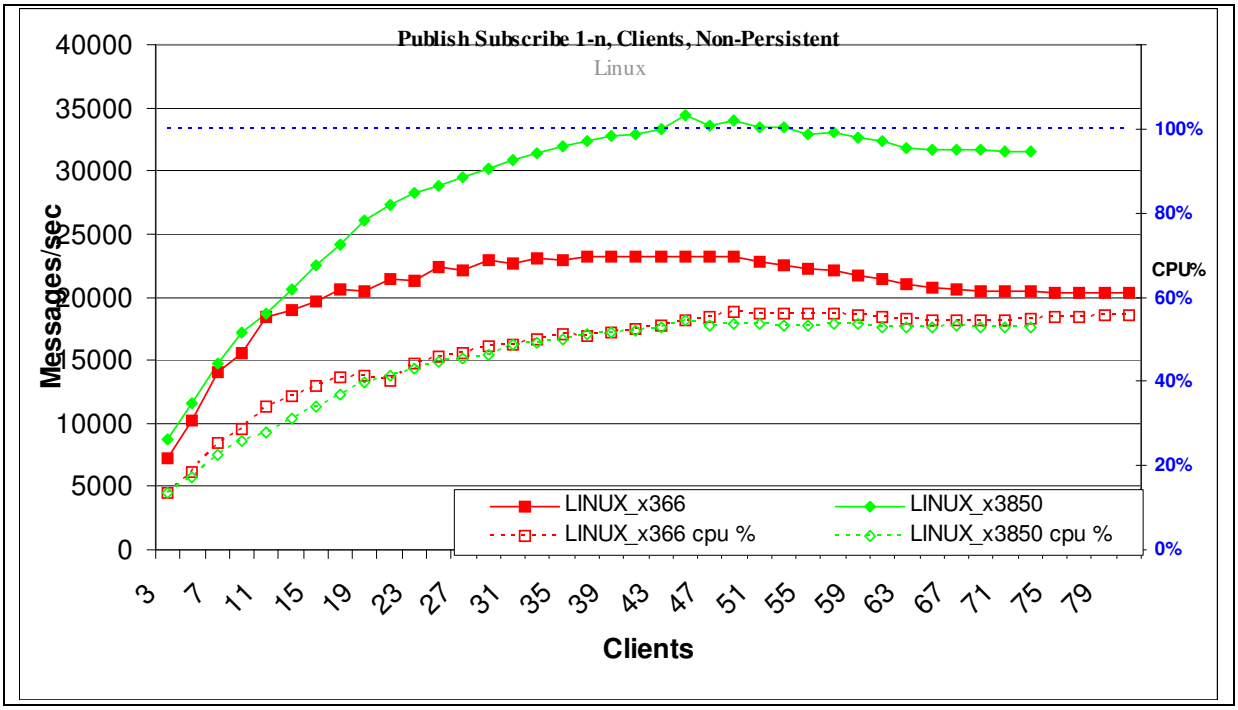

Figure 15 – Publish Subscribe 1:N, non persistent

Test name: **LPSN**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 2 subscribers Per publication**
Linux_x3850	**45**	**34439**	**765**	**54%**	**2940**
Linux_x366	**41**	**23263**	**567**	**52%**	**2401**

Table 10 – Publish/Subscribe 1:N, non Persistent messages

The publisher produces messages as fast as possible. Initially there are 2 subscribers and one publisher when 2940 publications per second can be achieved on Linux_x3850 and 2401 on Linux_x366. The response time for the publish command increases as the number of subscribers increase. On Linux_x3850 with 44 subscribers, the publisher creates 765 messages per second and 567 per second with 40 subscribers on Linux_x366.

4.1.2 Publish Subscribe 1:N, Non Persistent messages, Cores

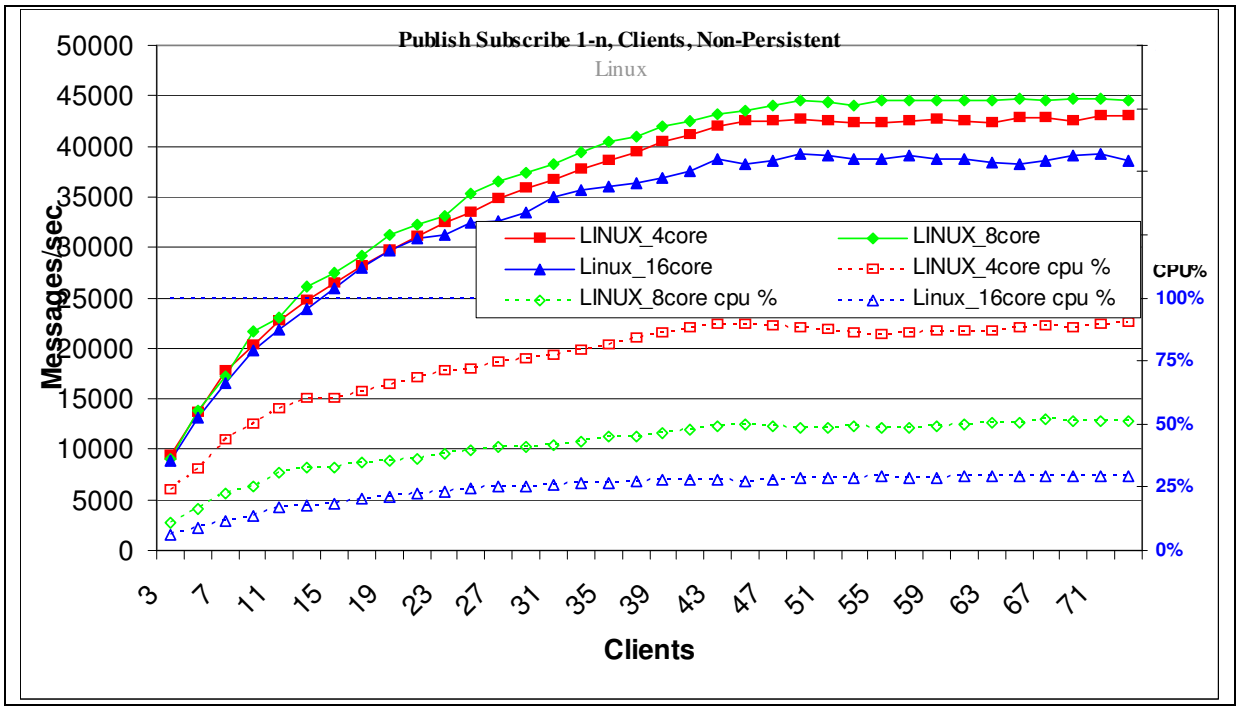

Figure 16 – Publish Subscribe 1:N, non persistent, Cores

Test name: **LPSN_cores**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 2 subscribers Per publication**
Linux_4core	**73**	**43000**	**589**	**54%**	**3106**
Linux_8core	**71**	**44534**	**627**	**51%**	**3037**
Linux_16core	**49**	**39258**	**501**	**29%**	**2967**

Table 11 – Publish/Subscribe 1:N, non Persistent messages

These multi core measurements used an X7350 with 16 cores of 2.93 GH. Cores are off-lined to produce the 4 core and 8 core machine. The 8 core system processes 3% more traffic than the 4 core. The 16 core system processes 7% less traffic than the 4 core system

The publisher produces messages as fast as possible. Initially there are 2 subscribers and one publisher when 2940 publications per second can be achieved on Linux. The response time for the publish command increases as the number of subscribers increase. On Linux with 44 subscribers, the publisher creates 765 messages per second which are all consumed by the subscribers.

Doubling the number of cores from 4 to 8 provides a performance gain of under 4% which suggests that 4 cores is the economic option. Doubling from 8 cores to 16 cores degrades throughput showing that using multiple queue mangers in a cluster on this single machine would be more efficient.

4.1.3 Publish Subscribe 1:N, Persistent messages

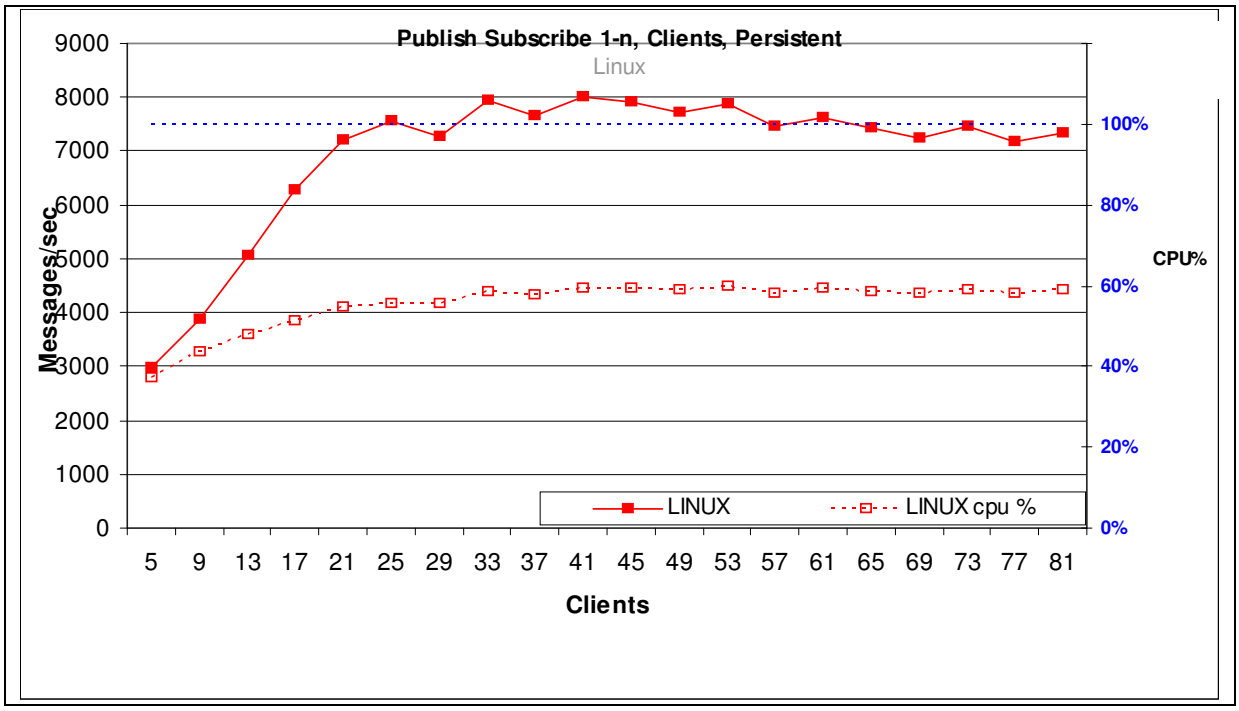

Figure 17 – Publish Subscribe 1:N, persistent

Test name: **LPSP**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 4 subscribers Per publication**
Linux_x3850	**21**	**7218**	**343**	**55%**	**597**

Table 12 – Publish/Subscribe 1:N, Persistent messages

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 597 publications per second can be achieved on Linux. The response time for the publish command increases as the number of subscribers increase. On Linux with 20 subscribers, the publisher creates 343 messages per second which are all consumed by the subscribers

4.1.4 Publish Subscribe 1:N, Non Persistent messages, Cluster-4

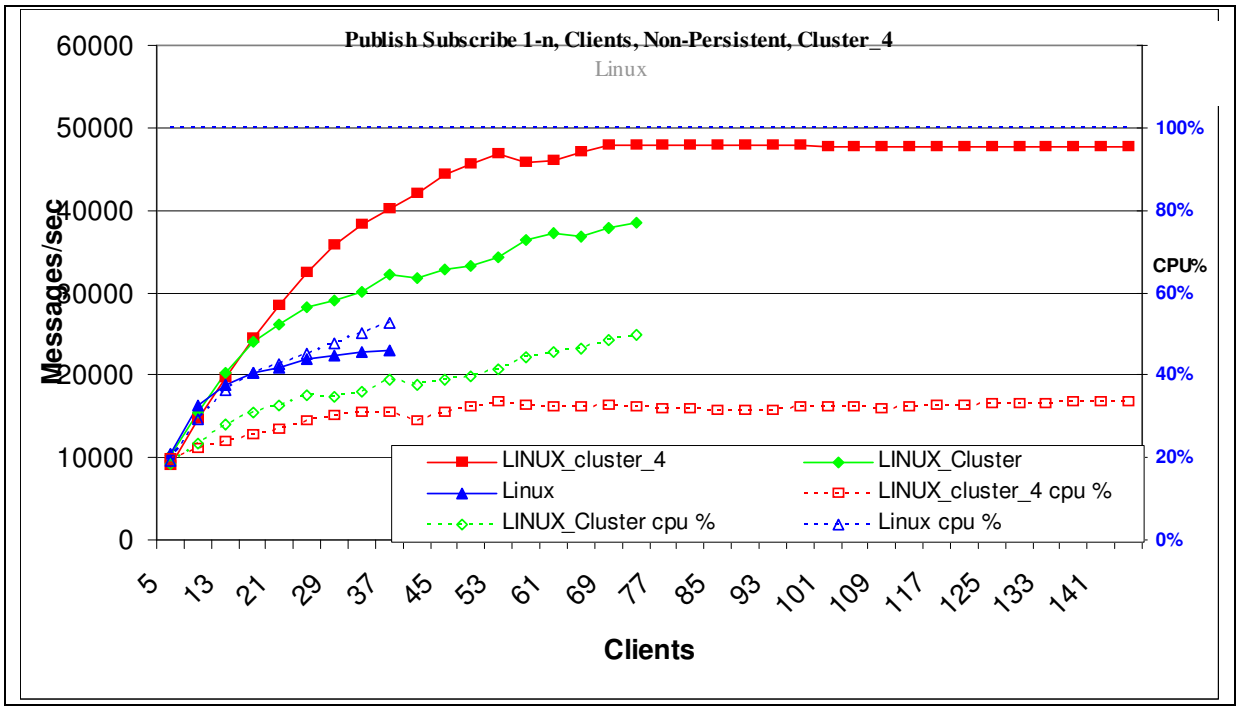

Figure 18 – Publish Subscribe 1:N, non persistent, Cluster

Test name: **LPSN_C4**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second Per publication With 'n' subs**			
				Pub	**Sub**	**n=4**	**n=8**	**n=12**	**n=16**
Linux_x366	**37**	**22920**	**619**	**52**		**2094**	**1736**	**1455**	**1213**
Linux Cluster-2	**73**	**38407**	**526**	**50**	**58**	**2029**	**1710**	**1562**	**1408**
Linux Cluster-4	**73**	**47928**	**656**	**32**	**36**	**1780**	**1630**	**1517**	**1433**

Table 13 – Publish/Subscribe 1:N, non Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 2044 publications per second can be achieved on Linux.
The initial 2 Queue Managers use x366 systems while the 3rd and 4th Queue Managers use x3850 systems. The second Queue Manager provides an increase of 67% in message throughput while the addition of the second pair adds an additional 25%.
There needs to be at least 8 subscribers per topic before a performance benefit is apparent from Clustering. . This is because of the overhead in transferring the publication to the alternate Queue Manger. At least 16 subscribers are needed before a performance benefit is seen from the second pair of Queue Managers. The server CPU is split into 2 parts, the original machine with Publisher and Subscribers(Pub) , together with the machine only hosting subscribers(Sub). They are identical machines and it can be observed that the Sub machine is 13%-16% busier than the Pub machine.

4.1.5 Publish Subscribe 1:N, Non Persistent messages, Cluster Publications

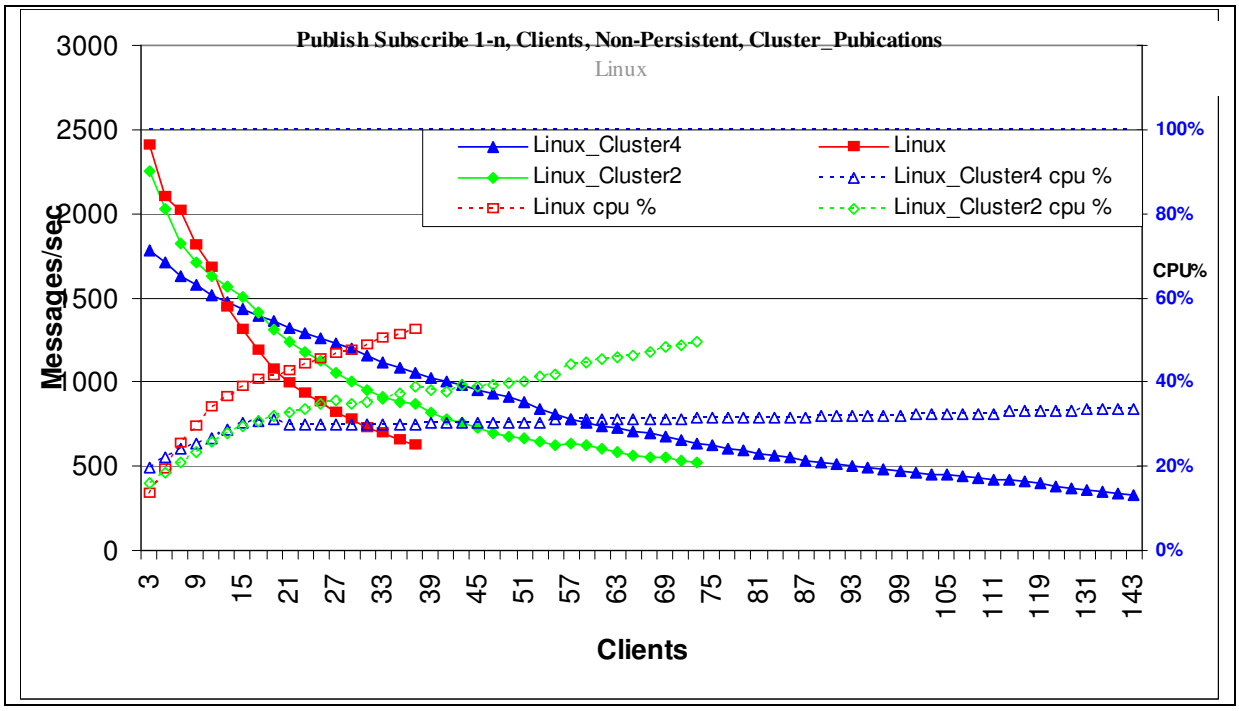

Figure 19 – Publish Subscribe 1:N, non persistent, Cluster publications

Test name: **LPSN_CP**	**Subscribers for 620**	**620 Pubs Per second**	**620 CPU%**	**Subscribers For 520**	**520 Pubs Per second**	**520 CPU%**
Linux_x366	**36**	**22940**	**52%**			
Linux Cluster_2	**58**	**35960**	**39%**	**72**	**37960**	**50%**
Linux Cluster_4	**74**	**46500**	**30%**	**88**	**46280**	**32%**

Table 14 – Publish/Subscribe 1:N, non Persistent messages, Cluster

The Graph is a different way of looking at the Message capacity of a system. It examines the effect of subscribers on the Publishing rate. A single publisher is publishing as fast as possible but the latency for each publication gradually increases which means the publication rate is gradually reduced as the number of subscribers increase. A system requirement to publish 620 a second can be dealt with by a single Queue manager with 36 subscribers, by 2 queue managers in a cluster for 58 subscribers or with 4 QM in a cluster for 74 subscribers. The second QM in a cluster increases the capacity by 56%. The system with 4 QM in the cluster provides 30% more messaging capacity than using 2 QM although the 3rd and 4th QM are using x3850 hardware.

4.1.6 Publish Subscribe 1:N, Persistent messages, Cluster

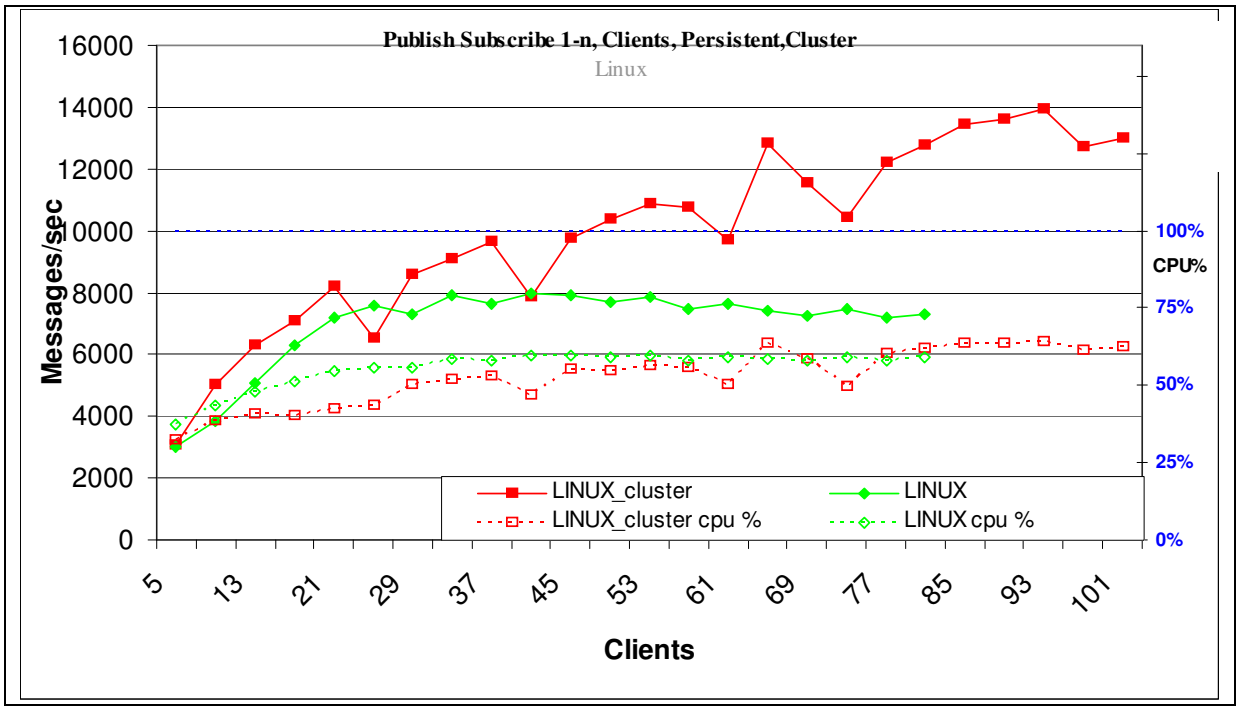

Figure 20 – Publish Subscribe 1:N, persistent, cluster

Test name: **LPSP_C**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second With 4 subscribers Per publication**
				Pub	**Sub**	
Linux_x3850	**41**	**7996**	**195**	**55%**		**597**
Linux Cluster	**53** **93**	**10979** **13910**	**205** **149**	**52%** **57%**	**57%** **64%**	**469**

Table 15 – Publish/Subscribe 1:N, Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 469 publications per second can be achieved on Linux cluster. The response time for the publish command increases as the number of subscribers increase. On Linux cluster with 52 subscribers, the publisher creates 205 messages per second and with 92 subscribers there are 149 publications per second. The throughput with a small number of subscribers is limited by the MQ log and it needs more than 6 subscribers before it is worthwhile using clustering. Clustering can increase the messages processed by 74% when using a second x3850. The Pub machine contains the publisher and subscribers. The Sub machine only contains subscribers in the clustered environment. The Sub machine is busier than the machine containing the Publisher.

4.1.7 Publish Subscribe (Multiple P/T/S), Non Persistent messages

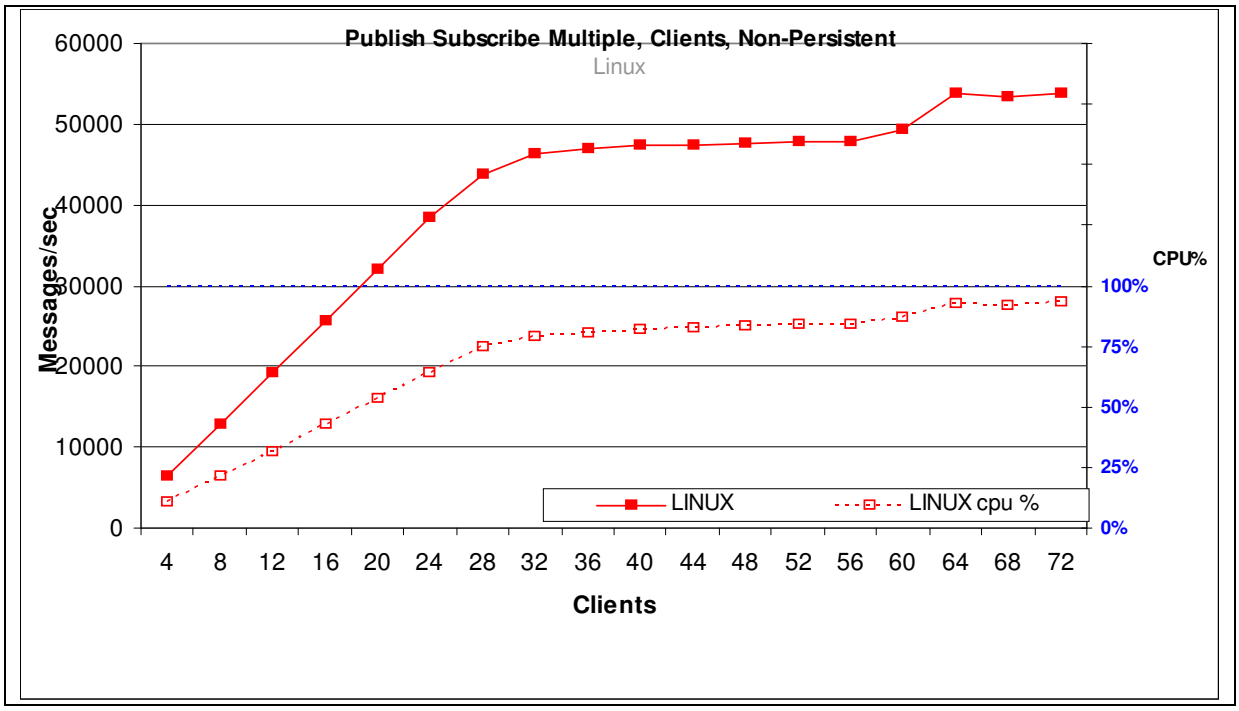

Figure 21 – Publish Subscribe Multiple, non persistent

Test name: **LPTN**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
Linux_x3850	**28**	**43867**	**1566**	**74%**
	64	**53737**	**839**	**93%**

Table 16 – Publish/Subscribe Multiple, non Persistent messages

Each publisher creates 1600 non-persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 14 producers and 14 consumers (28 Clients) on Linux the expected throughput is 1600*14*2=44800 whereas the measured throughput is 43867 messages per second. Additional publishers can increase the system capacity to 53737 messages per second

4.1.8 Publish Subscribe (Multiple P/T/S), Persistent messages

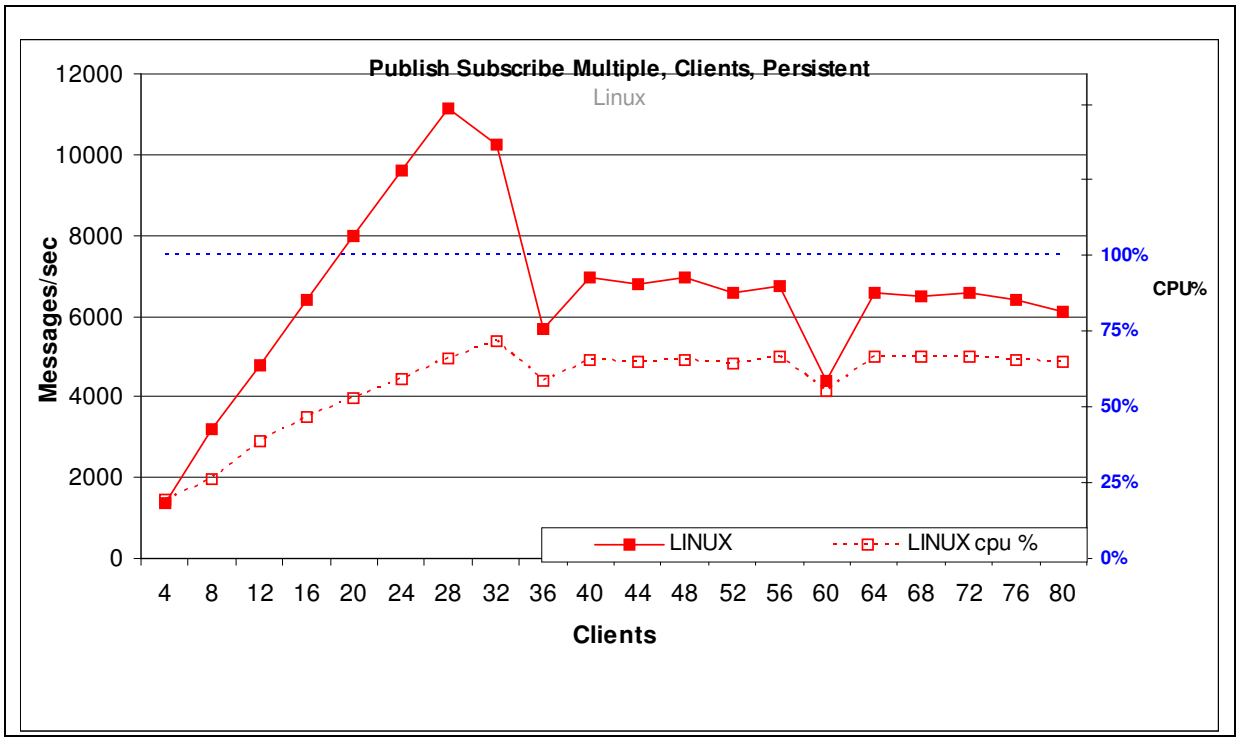

Figure 22 – Publish Subscribe Multiple, persistent

Test name: **LPTP**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
Linux_x3850	**28**	**11161**	**398**	**66%**

Table 17 – Publish/Subscribe Multiple, Persistent messages

Each publisher creates 400 persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 14 producers and 14 consumers (28 Clients) on Linux the expected throughput is 400*14*2=11200 whereas the measured throughput is 11161 messages per second. Additional clients cause the overall messaging rate to decline because the MQ Logger has become the bottleneck.

4.1.9 Point to Point (Multiple P/Q/C), Non Persistent messages

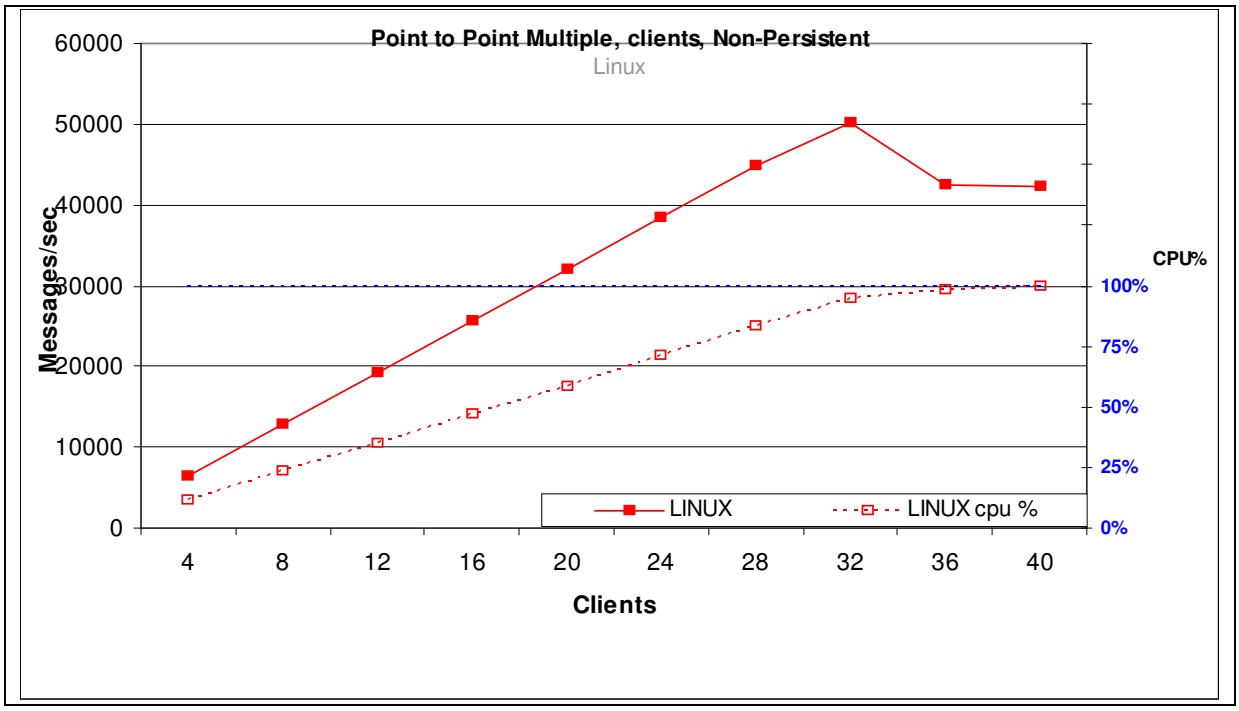

Figure 23 – Point to Point, Multiple, non persistent

Test name: **LTPN**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
Linux_x3850	**32**	**50259**	**1570**	**94%**

Table 18 – Point to point ,Multiple, non Persistent messages

Each publisher creates 1600 non persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 16 producers and 16 consumers (32 Clients) on Linux, the expected throughput is 1600*16*2=51200 whereas the measured throughput is 50259 messages per second. Additional clients cause the messaging rate to decline because the CPU is very busy.

4.1.10 Point to Point (Multiple P/Q/C), Persistent messages

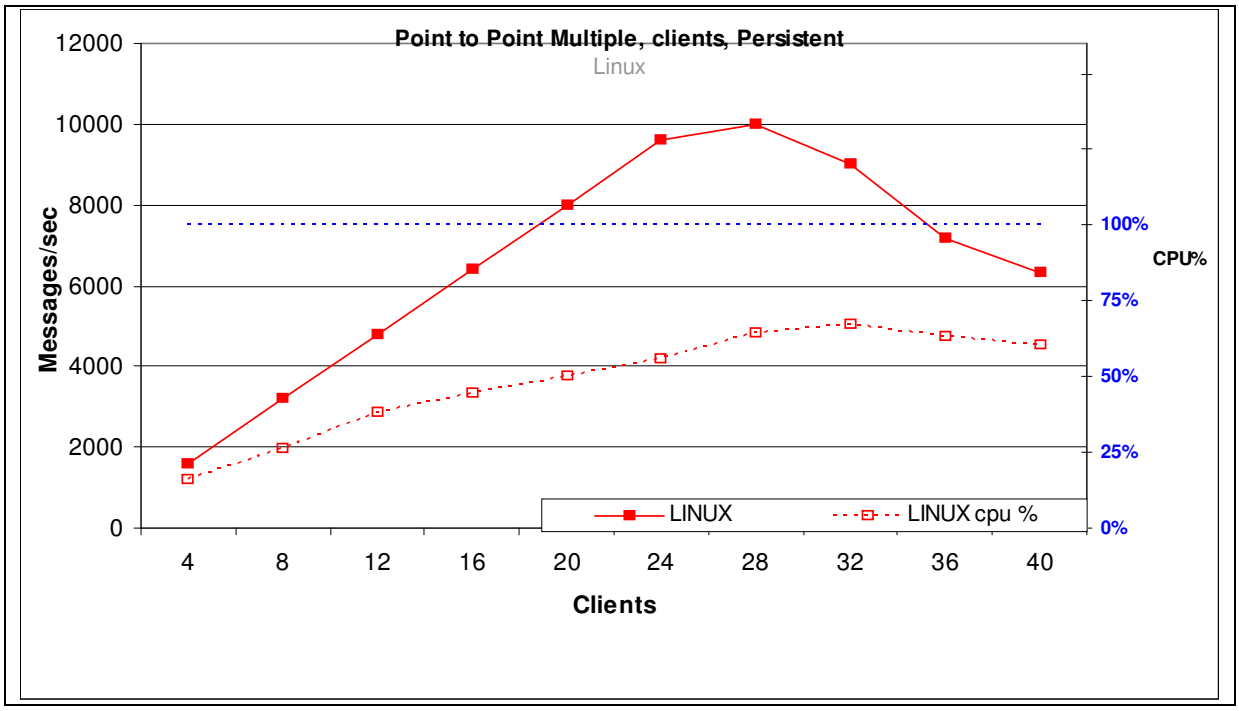

Figure 24 – Point to point, multiple, persistent

Test name: **LTPP**	**Clients**	**Messages Per second**	**Publication Per second**	**Server CPU**
Linux_x3850	**24**	**9591**	**399**	**56%**
	28	**10010**	**357**	**64%**

Table 19 – Point to point ,Multiple, Persistent messages

Each message producer creates 400 messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 12 producers and 12 consumers (24 Clients) on Linux, the expected throughput is 400*12*2=9600 whereas the measured throughput is 9591 messages per second. The other measurements of Linux Persistent messages in this chapter show that the logger capacity is maximised at 11000 messages per second.

5 Solaris Measurements

5.1.1 Publish Subscribe 1:N, Non Persistent messages

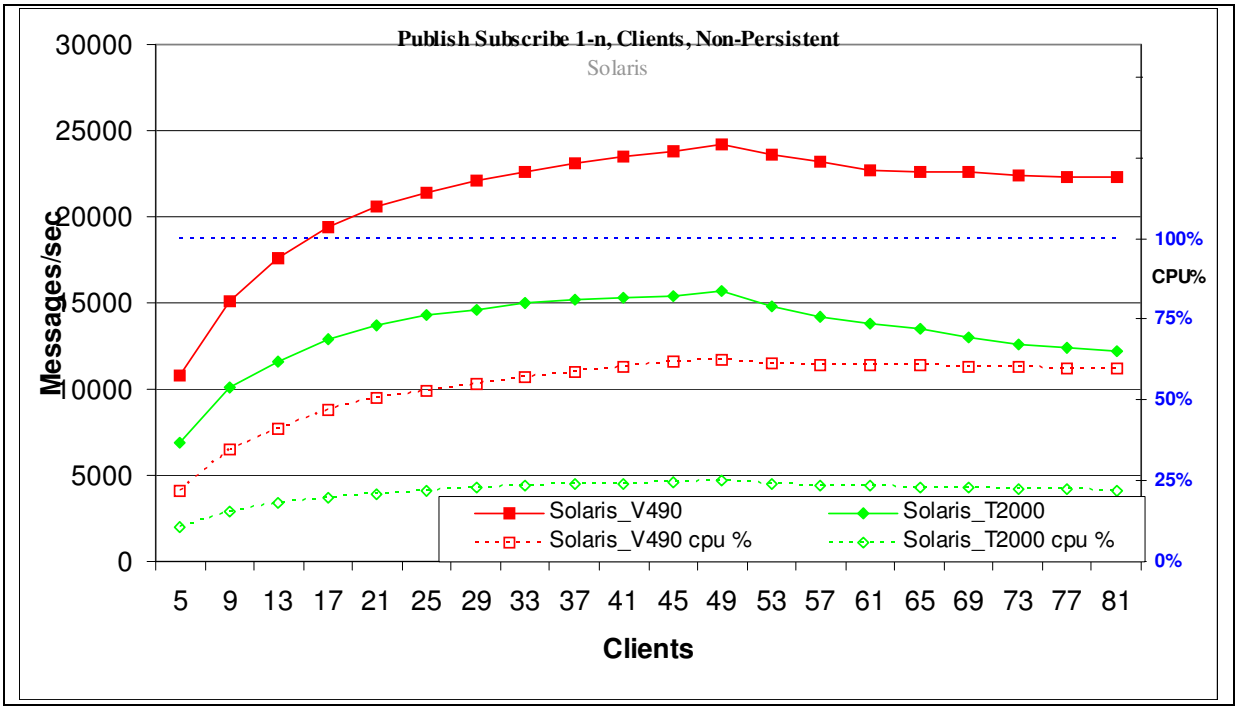

Figure 25 – Publish Subscribe 1:N, non persistent

Test name: **SPSN**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 4 subscribers Per publication**
Solaris V490	**49**	**24176**	**493**	**62%**	**2151**
Solaris T2000	**49**	**15731**	**321**	**25%**	**1379**

Table 20 – Publish/Subscribe 1:N, non Persistent messages

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 2151 publications per second can be achieved on Solaris V490 or 1379 on Solaris T2000. The response time for the publish command increases as the number of subscribers increase. On Solaris V490 with 48 subscribers, the publisher creates 493 messages per second and the Solaris T2000 publisher creates 321 per second which are all consumed by the subscribers. Additional subscribers do not increase the messaging capacity of the system. A comparison is shown of a T2000 and a V490 system whereas most measurements in this chapter use the V490.

5.1.2 Publish Subscribe 1:N, Persistent messages

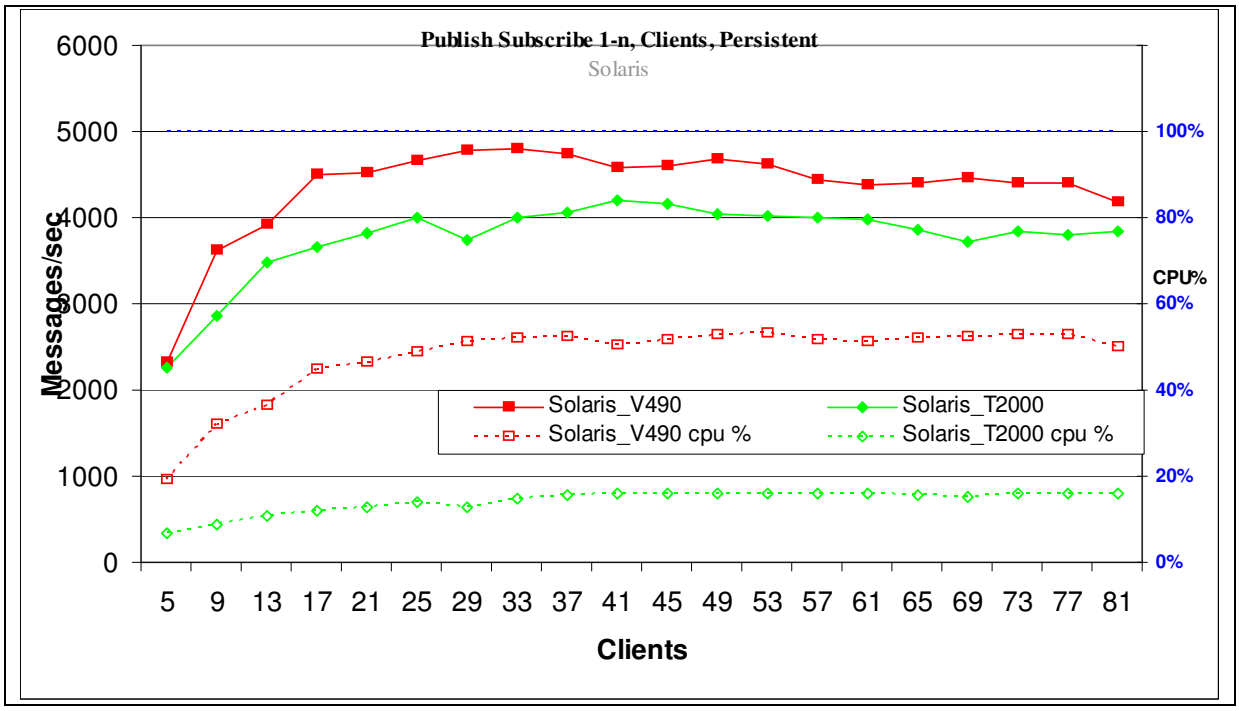

Figure 26 – Publish Subscribe 1:N, persistent

Test name: **SPSP**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**	**Pubs per second With 4 subscribers Per publication**
Solaris V490	**33**	**4801**	**145**	**52%**	**463**
Solaris T2000	**41**	**4205**	**102**	**16%**	**453**

Table 21 – Publish/Subscribe 1:N, Persistent messages

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 463 publications per second can be achieved on Solaris V490 or 453 on Solaris T200. The response time for the publish command increases as the number of subscribers increase. On Solaris V490 with 32 subscribers, the publisher creates 145 messages per second and on Solaris T2000, the publisher creates 102 messages per second which are all consumed by the subscribers. Additional subscribers do not increase the messaging capacity of the system. A comparison is shown of a T2000 and a V490 system.

5.1.3 Publish Subscribe 1:N, Non Persistent messages, Cluster

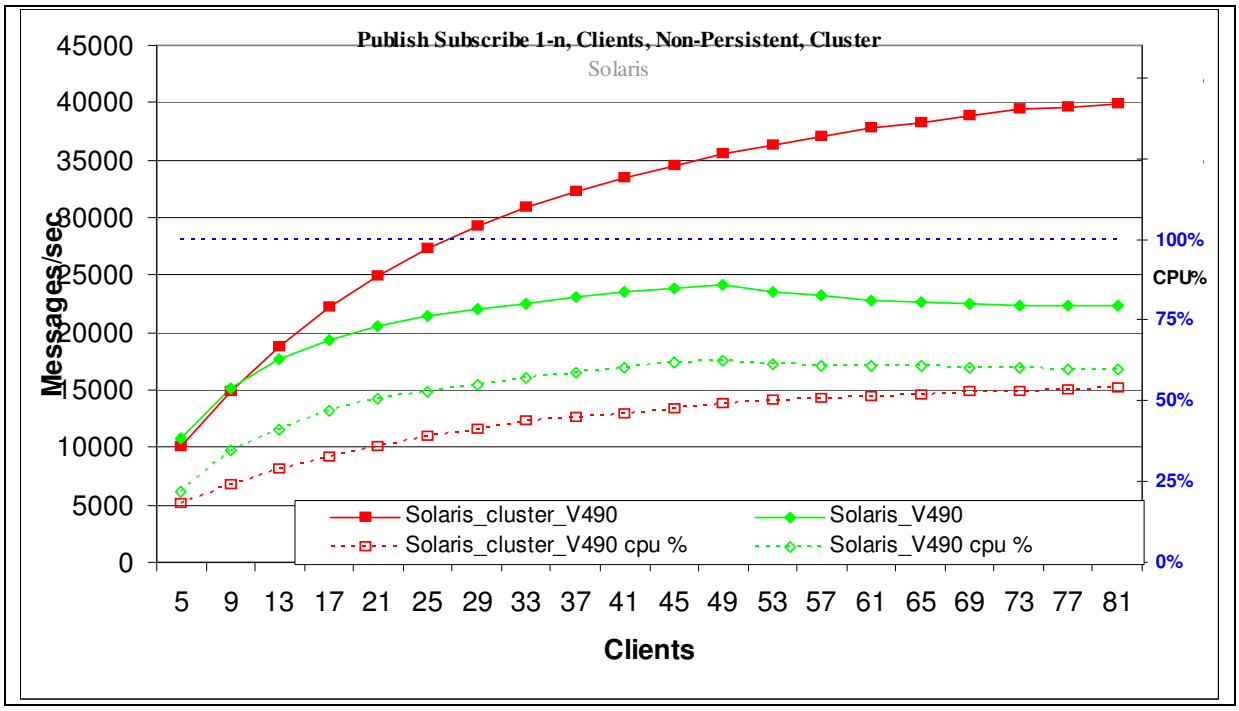

Figure 27 – Publish Subscribe 1:N, non persistent, Cluster

Test name: **SPSN_C**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second With 4 subscribers Per publication**
				Pub	**Sub**	
Solaris V490	**49**	**24176**	**493**	**62%**		**2151**
Solaris Cluster	**81**	**39938**	**493**	**54%**	**58%**	**2008**

Table 22 – Publish/Subscribe 1:N, non Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 2151 publications per second can be achieved on Solaris . Using a cluster of two identical V490s reduces the publication rate to 2008 because of the overhead involved of using the second Pub-Sub engine. The response time for the publish command increases as the number of subscribers increases and the single Pub-Sub engine can publish 493 messages per second with 49 clients whereas the clustered system can publish 493 messages per second to 81 clients. The second Queue manager has enabled the messaging rate to increase by 65%. There needs to be more than 10 subscribers to the topic before a performance benefit is apparent. Half the subscribers are attached to each Queue manager but the QM with the publisher attached uses a smaller CPU% in the cluster test.

5.1.4 Publish Subscribe 1:N, Persistent messages, Cluster

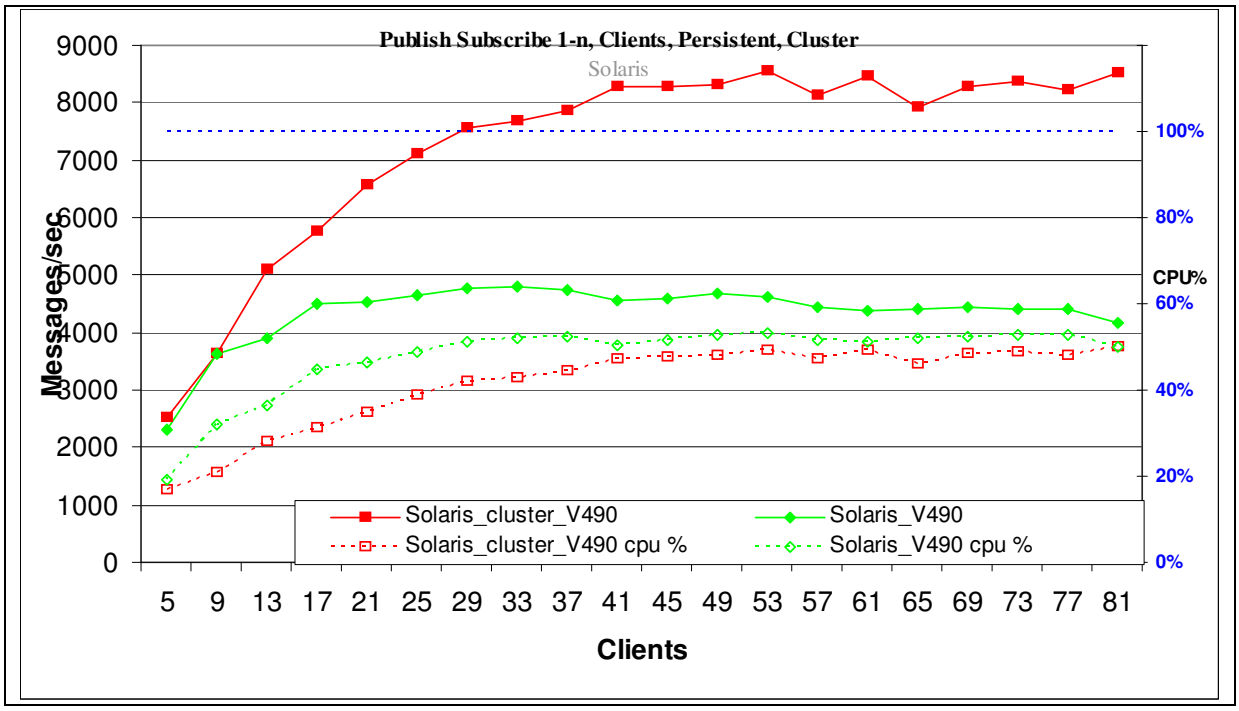

Figure 28 – Publish Subscribe 1:N, persistent, cluster

Test name: **SPSP_C**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**		**Pubs per second With 4 subscribers Per publication**
				Pub	**Sub**	
Solaris V490	**33**	**4801**	**145**	**52%**		**463**
Solaris Cluster	**53**	**8539**	**161**	**49%**	**49%**	**469**

Table 23 – Publish/Subscribe 1:N, Persistent messages, Cluster

The publisher produces messages as fast as possible. Initially there are 4 subscribers and one publisher when 463 publications per second can be achieved on Solaris with a similar number in a cluster. The response time for the publish command increases as the number of subscribers increase. On Solaris with 32 subscribers, the publisher creates 145 messages per second and a Solaris cluster publisher with 52 subscribers achieves 161 publications per second. The throughput with a small number of subscribers is limited by the MQ. Clustering can increase the messages processed by 77%. . Half the subscribers are attached to each Queue manager but the QM with the publisher attached uses a similar CPU% in the cluster test.

5.1.5 Publish Subscribe (Multiple P/T/S), Non Persistent messages

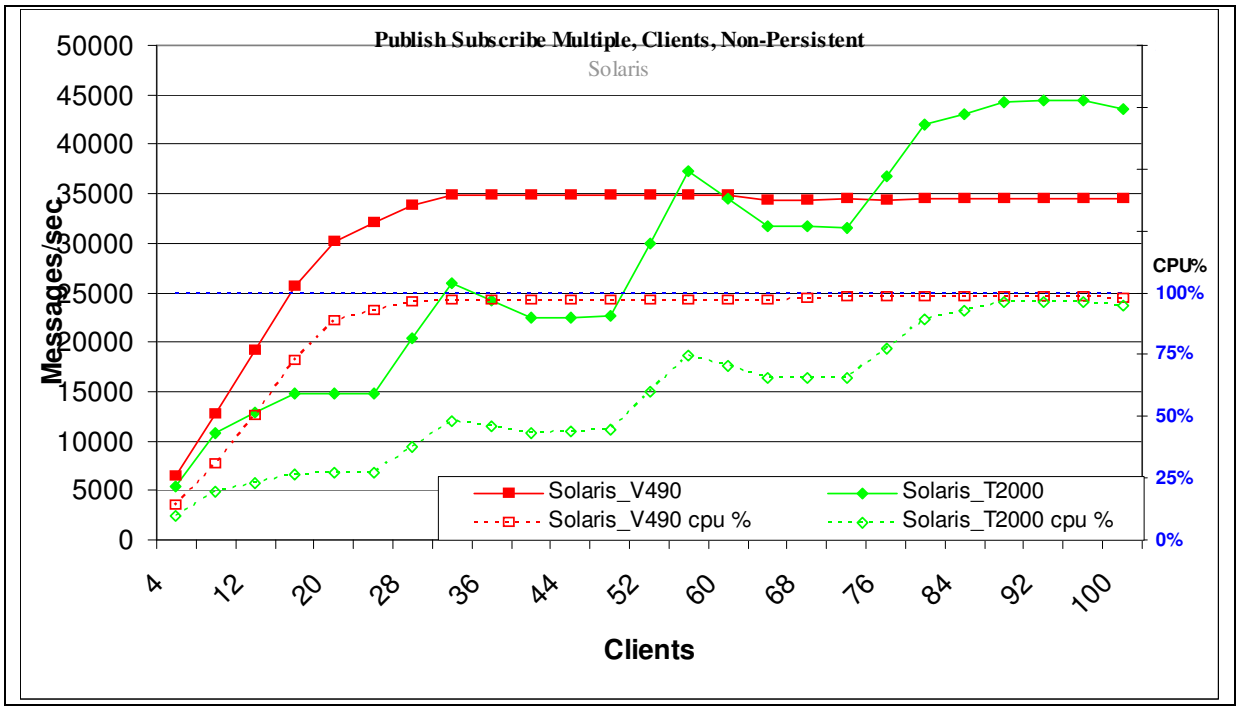

Figure 29 – Publish Subscribe Multiple, non persistent

Test name: **SPTN**	**Clients**	**Messages Per second**	**Publications per second**	**Server CPU**
Solaris V490	**20**	**30153**	**1507**	**88%**
	32	**34797**	**1087**	**97%**
Solaris T2000	**64**	**25965**	**405**	**48%**
	92	**44342**	**482**	**96%**

Table 24 – Publish/Subscribe Multiple, non Persistent messages

Each publisher creates 1600 non-persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 10 producers and 10 consumers (20 Clients) on Solaris V490 the expected throughput is 1600*10*2=44800 whereas the measured throughput is 43867 messages per second. The Solaris T2000 is an 8 core with 4 threads per core. The increase in throughput as the number of clients increases is not as smooth as the V490.

5.1.6 Publish Subscribe (Multiple P/T/S), Persistent messages

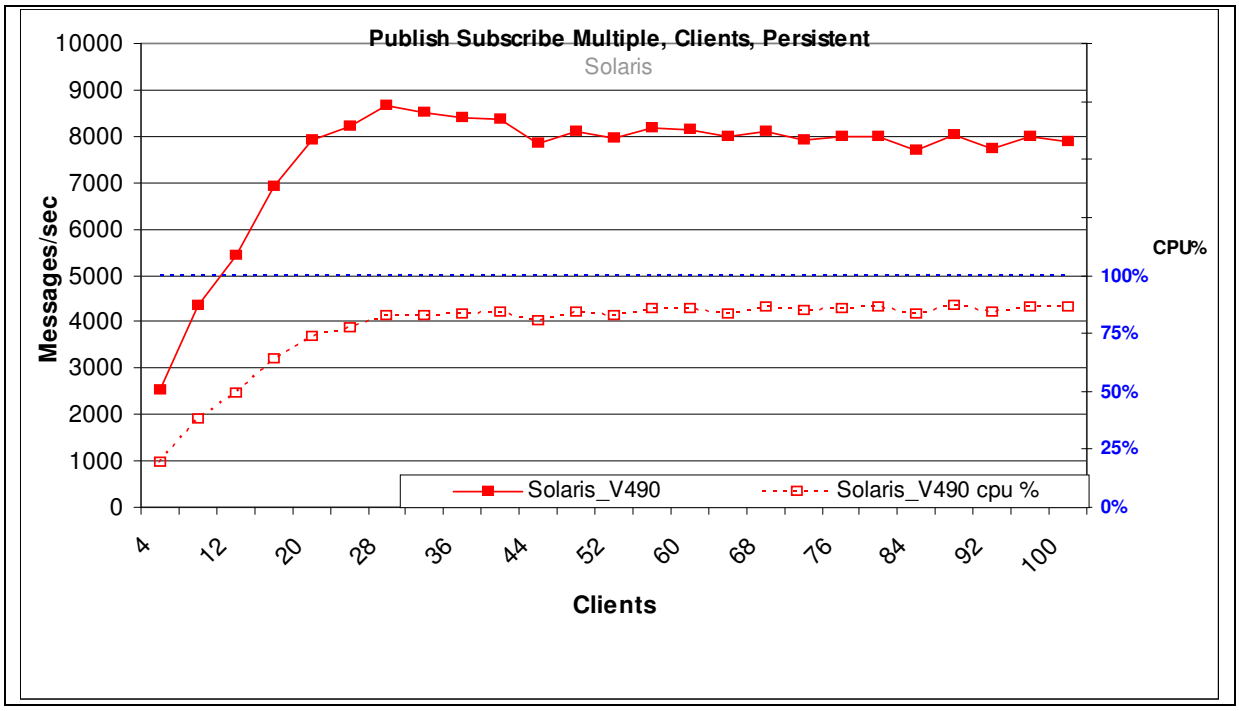

Figure 30 – Publish Subscribe Multiple, persistent

Test name: **SPTP**	**Clients**	**Messages Per second**	**Publications Per second**	**Server CPU**
Solaris V490	**28**	**8644**	**308**	**82%**

Table 25 – Publish/Subscribe Multiple, Persistent messages

Each publisher creates 400 persistent messages per second. With 14 Publishers and 14 Subscribers (28 Clients) on Solaris the expected throughput is 400*14*2=11200 whereas the measured throughput is 8644 messages per second.

5.1.7 Point to Point (Multiple P/Q/C), Non Persistent messages

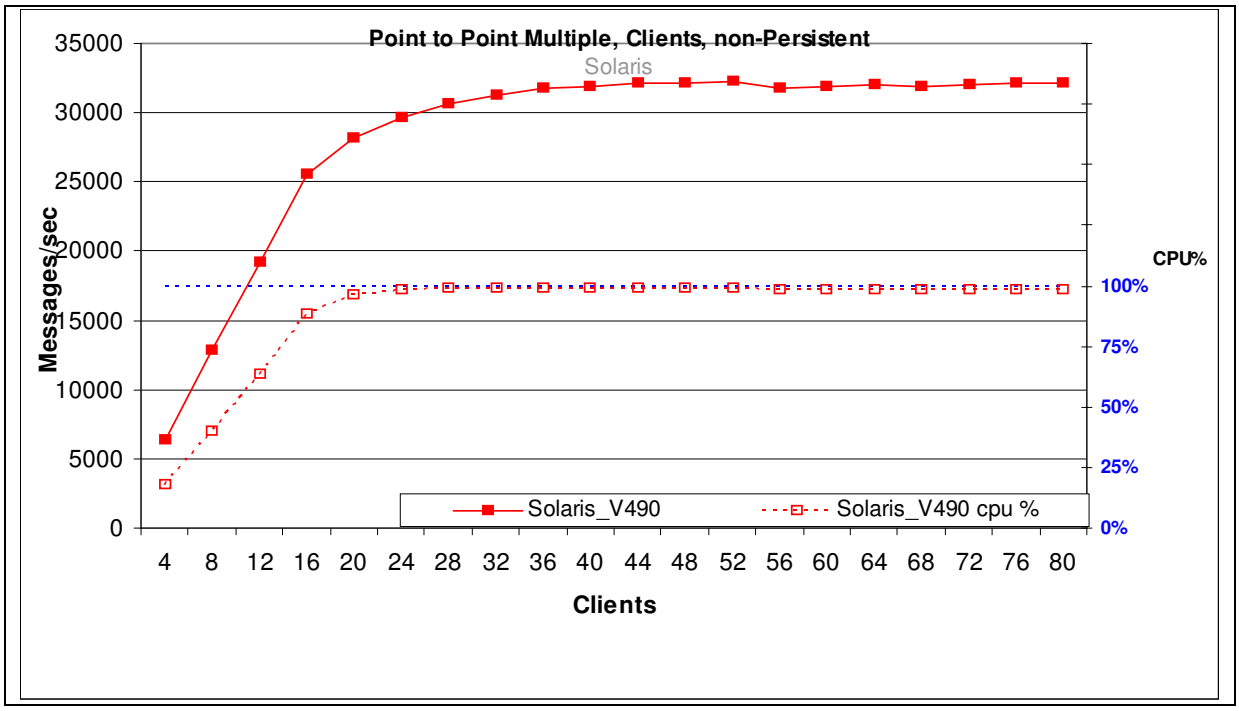

Figure 31 – Point to Point, Multiple, non persistent

Test name: **STPN**	**Clients**	**Messages Per second**	**Server CPU**
Solaris V490	**16**	**25489**	**88%**
Solaris V490	**28**	**30692**	**99%**

Table 26 – Point to point ,Multiple, non Persistent messages

Each producer creates 1600 non persistent messages per second and the system throughput increases as a straight diagonal line until the system capacity is achieved. With 8 producers and 8 consumers (16 Clients) on Solaris, the expected throughput is 1600*8*2=25600 whereas the measured throughput is 25489 messages per second. Additional message producers can increase the system capacity to 30692 messages per second

5.1.8 Point to Point (Multiple P/Q/C), Persistent messages

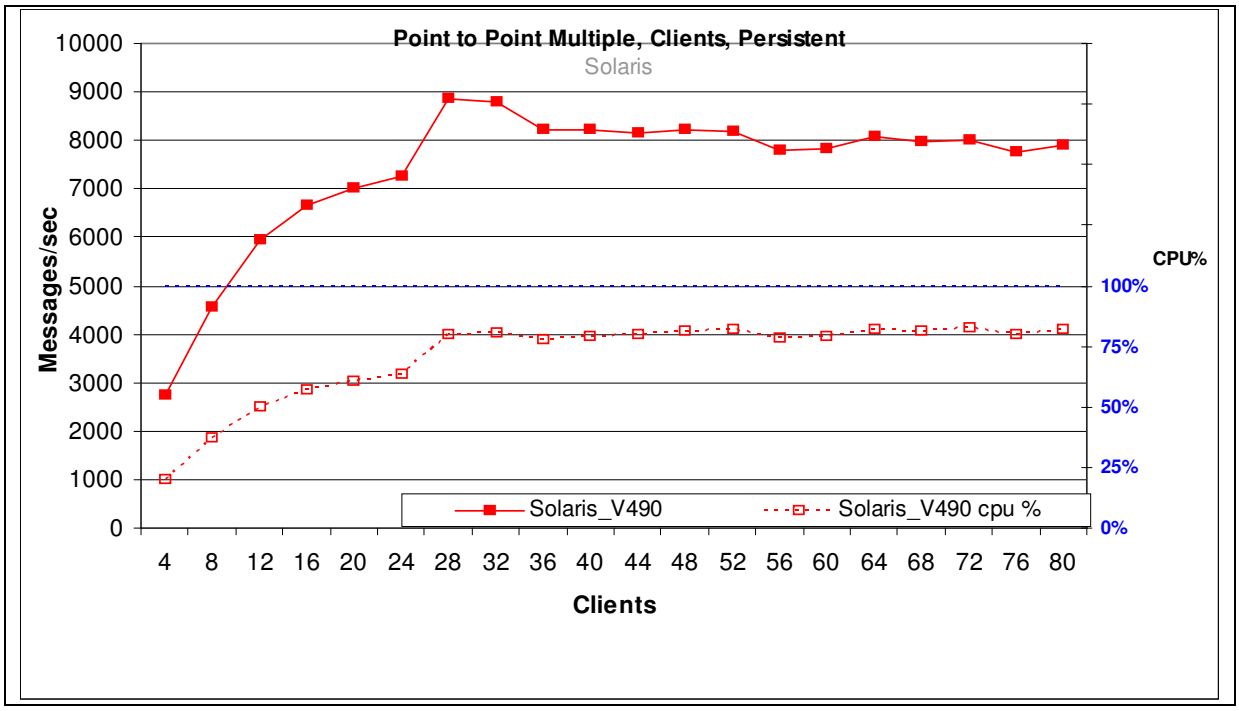

Figure 32 – Point to point, multiple, persistent

Test name: **STPP**	**Clients**	**Messages Per second**	**Server CPU**
Solaris V490	**28**	**8849**	**79%**

Table 27 – Point to point ,Multiple, Persistent messages

Each message producer creates 400 messages per second. With 14 producers and 14 consumers (28 Clients) on Solaris, the measured throughput is 8849 messages per second. Additional Producer/Consumers do not increase the system capacity

6 Machine and Test Configurations

The MQI applications used in this report to generate the performance data are Located on the Linux clients that communicate with the various server.

6.1 Linux, AIX, & Solaris Servers

Linux server machine (Red Hat 4.1.2-46 kernel 2.6.18-164.9.1.el5
IBM x3850: Model: x3850 M2 8864 4RG
Processor: 3.3GHz Intel xeon (7140N)
Architecture: 2 dual core CPU (4 way SMP)
Hyperthreading disabled
Memory (RAM): 4Gb
Disk: 2 Internal 16bit SCSI (90Gb each, 1 O/S, swap)
2 SAN disks on DS6000 (5Gb each, 1 queue, 1 log)
Network: 1Gbit Ethernet Adapter

Linux server machine(4.1.2 Multi core measurement) is
IBM x7350: Model:
Processor: 2.93GHz Intel xeon
Architecture: 16 CPU Hyperthreading disabled
Memory (RAM): 32Gb 4Mb cache
Disk: 4 Internal 16bit SCSI (70Gb each, 1 O/S, swap)
Network: 1Gbit Ethernet Adapter

AIX server machine

IBM p570: Processor: 4.2GHz Power 6
Architecture: 4 dual core cpu (8 core) .
Memory (RAM): 16Gb
Disk: 2 Internal 16bit SCSI (9Gb each, 1 O/S, swap)
2 SAN disks on DS6000 (5Gb each, 1 queue, 1 log)
Network: 1Gbit Ethernet Adapter

IBM Power 5 (used in cluster measurements)

System Model: IBM,9117-570

Processor Type: PowerPC_POWER5

Number Of Processors: 8

Processor Clock Speed: 1654 MHz

Memory Size: 15360 MB

Solaris server machines

Sun T2000: Model: T20|108C-64GA2G SFT2000
Processor: 1.4GHz
Architecture: 8core
Memory (RAM): 64Gb
Disk: Internal disks
Fibre channel HBA PCI-X 4Gb FC Dual Port HBA
2 SAN disks on DS6000 (5Gb each, 1 queue, 1 log)
Network: 1Gbit Ethernet Adapter

Sun V490:Processor: SPARCV9 @ 1500 HMz
Architecture: 4 CPU
Memory (RAM): 32Gb
Disk: Fibre channel HBA PCI-X 4Gb FC Dual Port HBA)
SAN with 2 partitions of 5Gb each

Network: 1Gbit Ethernet Adapter

The server operating system is

- Red Hat Enterprise Linux AS release 4 (Nahant Update 8) (kernel 2.6.9 -78.ELsmp)
- AIX 6.1
- SunOS 5.10 Generic_118833-36 sun4u sparc SUNW,Sun-Fire-V490
- SunOS 5.10 Generic_138888-03 sun4v sparc SUNW,Sun-Fire-T200

Clients are hosted by two 64 bit RHEL driver machines which is designed to ensure the server is the bottleneck.

Linux Drivers machines (Red Hat 4.1.2-46 kernel 2.6.18-164.9.1.el5) for AIX and Solaris measurements
IBM x3850: Model: x3850 M2 8864 4RG
Processor: 3.3GHz Intel xeon (7140N)
Architecture: 2 dual core CPU (4 way SMP)
Hyperthreading disabled
Memory (RAM): 4Gb
Disk: 2 Internal 16bit SCSI (90Gb each, 1 O/S, swap)
2 SAN disks on DS6000 (5Gb each, 1 queue, 1 log)
Network: 1Gbit Ethernet Adapter

Linux Driver machines for Linux server measurements
IBM x366: Model:
Processor: 3.66GHz Intel xeon
Architecture: 4 CPU Hyperthreading disabled
Memory (RAM): 8Gb 1Mb level 2 cache
Disk: 2 Internal 16bit SCSI (70Gb each, 1 O/S, swap)
2 SAN disks on DS6000 (5Gb each, 1 queue, 1 log)
Network: 1Gbit Ethernet Adapter

6.2 SAN disk subsystem

MQ Log and Queues on SAN disks on DS6000. 5GB allocated for both Log and Queues on Linux, Solaris and AIX machines.

The MQ SAN consists of a pair of 2026 model 432 (McDATA ES-4700) switches running at 4Gb/s with 32 ports each. They are connected together via two inter-switch links to form a single SAN fabric.

The MQ hosts attach via this SAN to a DS6800 disk array (1750 model 511) with one expansion drawer.

Each drawer (controller + expansion) contains 16 x 73Gb 15K fibre channel disk drives, so there are a total of 32 physical drives.

The 32 drives are configured as four RAID-5 arrays, each of which is 6+Parity+Spare (the number of spares is defined by the configuration of the DS6800).

The controller has an effective cache size of 2.6Gb plus 0.3Gb of NVS

6.3 Test case names

The first character defines he operating system (A***, L***,S***) represent AIX Linux and Solaris.
The second and third character pairs (*PS*, *PT*, *TP*) represent publish_subscribe 1:N, publish_subsciber_multiple, and Point to Point.
The forth character (***P, ***N) represents Persistent and Non-persistent messages.

7 Summary

7.1 Publish Subscribe 1:N

This scenario does not use more than 62% of a 4 way because the main Publishing thread will use one engine. Upgrading from a single Queue Manager to a clustered system improves the system capacity by between 50% and 70% for non-Persistent messages and by 70% to 90% for Persistent. The performance benefit of clustering is apparent when there are more than 20 subscribers to the topic.

The addition of a second Queue Manager into the cluster means that in the case of a Single Publisher, Multiple subscribers, there are only subscribers attached to the second Queue Manager. The first QM (labelled Pub) has to decide which publications are sent on the channel to the second QM (labelled Sub). The second QM receives messages over the channel and propagates them to the subscribers. An observation from the measurements is that the second QM (Sub) uses more CPU than the first QM (Pub).

7.2 Publish Subscribe (Multiple P/T/S)

This non-persistent scenario uses between 93% - 97% of a 4 way because each triplet (P/T/S) is independent of the other applications. Using Clustering when the subscriber is on a different Queue Manager to the publisher causes a degradation of up to 75% because there is only one subscriber to the topic. There are measurements of this clustered environment in this report.

8 Tuning

Performance reports with tuning information for WebSphere MQ v7.0 on all supported operating systems can be found via. the IBM SupportPac webpage at the following URL:
http://www.ibm.com/software/integration/wmq/support/
The main tuning actions taken for the tests in Chapter 2, 4, and 5 of this report were:
• 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)
• LogWriteIntegrity=SingleWrite (suitable for write-cached disks)
• Channels / MQIBindType = FASTPATH (channels are an extension to QM address space)
• TuningParameters / DefaultQBufferSize = 1MB (use 1MB of main memory per Q to hold non persistent messages before spilling to the file system
• TuningParameters / DefaultPQBufferSize = 1MB (use 1MB of main memory per Q to hold persistent messages)