MQSeries Publish/Subscribe Performance

24th March, 1999

MQSeries Performance Group
IBM United Kingdom Laboratories
Hursley Park
Hursley
Hants, SO21 2JN
United Kingdom

Notices

Information contained in this report has not been submitted to any formal IBM test and is distributed “as is”. The use of this information and the implementation of any techniques is the responsibility of the customer, and depends on the customer’s ability to evaluate and integrate them into their operational environment. While the information may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar resuts will be obtained elsewhere.

The performance data contained in this report was determined in a controlled environment and therefore the results obtained in other operating environments may vary significantly.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

- AIX
- MQSeries

The following terms are trademarks of other companies:

UNIX is a registered trademark in the United States and other countries licen sed exclusively through X/Open Company Limited.

Preface

This document presents the results from five different tests. The purpose of the first four evaluations was to ensure that the performance of the Publish/Subscribe code was in line with expectations founded on an understanding of base MQSeries performance characteristics. The final test attempted to duplicate a scenario planned for implementation by a Bank. Its purpose was to demonstrate that the broker performance would satisfy their requirements.

Summary of changes

26th March, 1999 Publish as SupportPac

Notices ii
Preface iii
Summary of changes iv
Executive Summary 1
Measurement Environment 2
Hardware configuration 2
Software configuration 2
Single Broker with a variable number of subscribers 3
The Genuine Broker test 3
The Simulated Broker test 3
Results 4
Conclusions 5
Single Broker with a variable number of subscribers and topics 6
Results 7
Conclusions 8
Single Broker with a variable number of subscriber queues 9
Results 10
Conclusions 11
“Loopback test” with two brokers 12
The Genuine Broker test 12
Simulated Broker test 12
Round trip using “Queue Magic” 12
Results 13
Conclusions 14
Bank scenario 16
Results 16
Conclusions 16

Executive Summary

The conclusions which can be drawn from the five evaluations described in this document are summarised below:

- We were able to achieve double the throughput rate initially required by the bank using a configuration similar to the one being implemented by them.

- When using persistent messaging (messages are logged to enable recovery) performance is very much dependent on having an optimal disk configuration. The queue and log files for each queue manager should be placed on fast, dedicated disks.

- The performance overhead of the broker is very small when using one or two brokers. Scaling beyond two brokers was not evaluated.

- Applications should be designed to share a common subscriber queue and retrieve messages using correlation-id. This is preferable to having a dedicated queue for each subsciber.

- If the application design dictates that many queues must be used (greater than approximately 128) then consider reviewing the broker parameter: *OpenCacheSize.*

Measurement Environment

Hardware configuration

The tests in this document were all performed using a single RISC/6000:

Model	Proc	MHz	L1 Cache (KB)	L2 Cache (MB)	Rel OLTP Perf	SPEC int_ rate95	SPEC int_ base_ rate95	SPEC fp_ rate95	SPEC fp_ base_ rate95
J50	604e/8	200	32/32	2	30.6	509	445	332	320

RS/6000 J50 8-Way with:

- 1 GB RAM
- 5 Fast/Wide SCSI Disks
- 4 SSA Disks
- 16 MBPS Token Ring Adapter

Software configuration

The RS/6000 was running with AIX 4.3.0.

Single Broker with a variable number of subscribers

This evaluation compared the results from two different types of test. The first type of test used the Publish/Subscribe function to distribute messages to a variable number of subscribers. The second type of test used regular MQSeries PUTs and GETs to achieve the same effect. By comparing the results, the costs of the broker function can be isolated.

The Genuine Broker test

For this type of test an application was used which subscribed to a single topic on a single stream queue a variable number of times (from 1 to 128). It then published a message with the appropriate topic and retrieved publication messages by correlid for each subscription. The time from publication until retrieval of the last response message was measured. By repeating the process an average figure was obtained for both persistent and non persistent messages of various sizes.

The tests were repeated using the Broker qm.ini option "*SyncPointIfPersistent=yes*". If this is specified, when the broker reads a publish or delete publication message from a stream queue during normal operation it specifies MQGMO_SYNCPOINT_IF_PERSISTENT. This makes the Broker receive non persistent messages outside syncpoint. If the broker receives a publication outside syncpoint, it will also forward that publication to subscribers outside syncpoint. The performance of non persistent messaging should improve when specifying yes to this option.

The Simulated Broker test

For this type of test an application was used which MQPUT a message to a stream queue (simulating publication) and then issued MQGET with wait in order to retrieve each response message. A second program simulated the function of a broker by retrieving the message from the stream queue and then issuing the required number of MQPUTs in reply.

Results

Message Size	Subscribers	Persistent Simulated	Persistent Broker	Non-Pers Simulated	Non-Pers Broker	NP, SPIP[1] Simulated	NP, SPIP Broker
10	1	32.11	35.22	5.76	6.02	4.58	4.78
10	2	44.98	47.46	6.3	6.67	5.9	5.52
10	4	67.67	67.99	7.93	8.33	6.88	7.27
10	8	106.77	110.14	11.01	11.51	10.14	10.67
10	16	204.64	202.97	17.2	18.78	16.95	16.59
10	32	366.44	369.85	29.6	32.14	28.76	32.75
10	64	702.3	704.39	56.15	57.5	52.03	62.44
10	128	1,376.26	1,384.23	125.36	122.16	102.03	106.19
100	1	33.88	34.68	5.66	6.49	4.27	5.04
100	2	45.34	*46.91*	6.44	6.81	5.05	5.53
100	4	68.39	68.28	7.97	*8.19*	6.65	7
100	8	109.49	110.77	11.34	11.68	10.02	10.19
100	16	206.78	204.25	17.74	18.27	16.64	16.51
100	32	372.09	378.16	30.1	31.56	28.94	29.63
100	64	710.48	714.79	58.61	62.3	54.22	54.79
100	128	1,407.99	1,409.97	118.32	128.74	105.34	109.62
1,000	1	37.8	39.2	5.69	6.28	4.28	5.2
1,000	2	53.02	55.08	6.83	6.98	5.12	5.73
1,000	4	77.13	76.17	8.12	9.14	6.68	7.54
1,000	8	124.42	130.06	11.69	12.04	10.05	10.7
1,000	16	217.04	219.61	19.26	18.51	16.6	17.54
1,000	32	398.4	409.03	30.82	34.42	30.28	34.61
1,000	64	774.55	779.09	68.65	70.74	61.56	64.08
1,000	128	1,547.52	1,571.39	139.87	155	119.82	120.41

The figures in this table are round-trip elapsed times measured in milliseconds.

SPIP *SyncPointIfPersistent=yes*

NP *Non persistent*

[1] Non Persistent with the “SyncPointIfPersistent” option

Conclusions

Comparing the simulated figures with those of the genuine broker we can see that the broker overhead is very small. For persistent messages the difference is generally less than 2 percent and within normal measurement variation. For non persistent messages the variation is also small as illustrated in the chart below. The "SyncPointIfPersistent" option does consistently improve elapsed times for non persistent messaging.

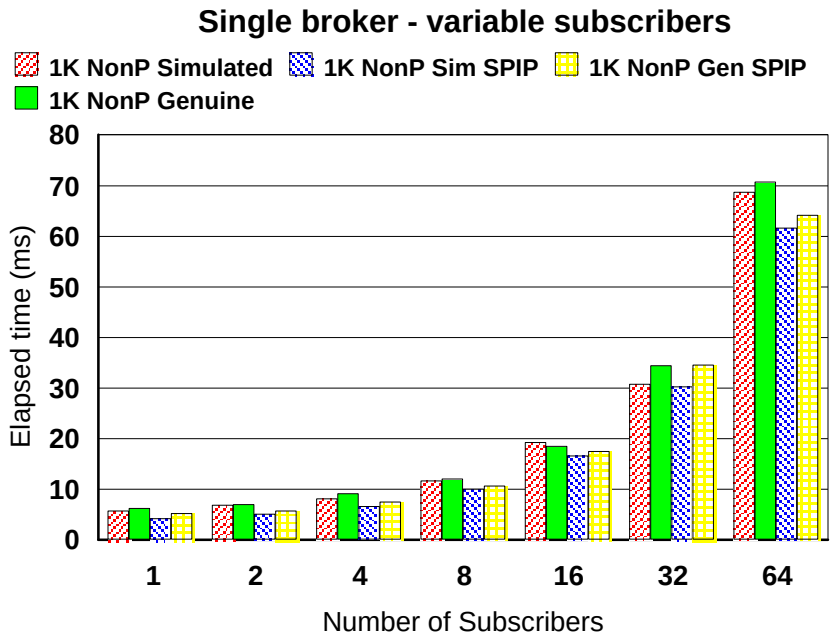


Chart 1: Elapsed time to deliver 1K non persistent messages to a varying number of subscribers using both genuine and simulated brokers - with and without SyncPointIfPersistent.

Single Broker with a variable number of subscribers and topics

This evaluation used the same applications as those described in the previous section . Its purpose was to investigate the effect on performance of varying the numbers of subscribers and topics.

This test differed from that in the previous section in that additional “dummy” subscribers were added. The application allows these dummy subscribers to subscribe to a single topic or to have each subscribed to a unique topic. These subscribers were ”dummies” in that their topics were chosen to be different fr om that used in any publication.

A small message size of ten bytes was chosen since the test is aimed at measuring the broker function and not that of the queue manager.

Results

Genuine subscribers	Extra dummy subscribers	Results with extra subs on single topic	Results with extra subs on multiple topics
1	0	4.57	
1	10	4.65	4.55
1	100	4.65	4.66
1	1,000	4.63	4.63
1	10,000	4.76	4.71
2	0	5.47	5.58
2	10	5.5	5.55
2	100	5.62	5.66
2	1,000	5.46	5.59
2	10,000	5.57	
4	0	7.15	
4	10	7.17	
4	100	7.31	
4	1,000	7.16	
4	10,000	7.6	
8	0	10.6	
8	10	10.24	
8	100	10.08	
8	1,000	10.06	
8	10,000	10.01	
16	0	16.52	
16	10	16.75	
16	100	16.71	
16	1,000	16.56	
16	10,000	16.64	
32	0	30.74	
32	10	29.81	
32	100	30.87	
32	1,000	29.8	
32	10,000	32.67	
64	0	55.82	
64	10	59.93	
64	100	71.36	
64	1,000	56.06	
64	10,000	55.53	
128	0	111.52	
128	10	108.6	
128	100	116.19	
128	1,000	107.75	
128	10,000	113.68	

Timings are in milliseconds

Conclusions

The results show that the elapsed time to deliver messages is not effected by the total number of subscriptions in the system or by the number of topics but is dependent on the number of messages actually being delivered. Subscriptions are maintained in a hashed table. The hashing prevents elapsed times increasing in line with the number of subscribers by removing the need to scan the table.

Single Broker with a variable number of subscriber queues

The purpose of this evaluation was to understand the effect on performance of having multiple subscriber queues rather than a single queue for all subscribers. The application used was the same as that described in "Measurement Environment".

The problem with test is that we need to determine the elapsed time from the point at which the publication is made until the message becomes available to an individual subscriber. With multiple subscribers we could have written a threaded application which captured the time at which messages were retrieved by each subscriber. Instead, we chose to use the simpler approach of publishing 15 times and timing how long it took to retrieve all 15 publications from a single subscribers perspective. The broker delivers all messages for a particular publication before moving on to a subsequent publication. We can therefore guarantee that at the time of completion for our monitored subscriber , at least 14 of the published messages have been delivered to the other subscribers .

The evaluation used non persistent messages with the *SyncPointIfpersistent* option turned on. Since we published 15 messages and then retrieved them, the times in this test are not directly comparable to those in the previous sections.

The results of this test were likely to be affected by the size of the cache maintained by the broker for recently used queues. The size of this cache is determined by the ***qm.ini*** parameter "*OpenCacheSize*" which was allowed to default to 128.

Results

Message Size	Subscribers	Single Queue	Multiple Queues
10	1	2.69	2.52
10	5	6.31	7.26
10	10	10.76	11.89
10	20	20.06	22.51
10	30	28.76	31.99
10	40	37.66	41.88
10	60	57.74	60.55
10	80	73.45	83.04
10	100	90.45	106.91
10	120	108.45	150.44
10	140	126.13	237.99
10	160	143.84	348.23
10	180	162.37	501.84
10	200	180.49	576.76
10	220	201.61	635.84
10	240	223.85	725.68
10	260	232.79	
10	280	249.93	
10	300	273.42	
1,000	1	2.76	2.80
1,000	5	6.66	14.63
1,000	10	11.58	24.63
1,000	20	22.29	53.85
1,000	30	32.43	67.92
1,000	40	42.43	82.53
1,000	60	62.67	125.96
1,000	80	81.59	159.03
1,000	100	100.73	182.64
1,000	120	128.19	241.00
1,000	140	140.96	313.60
1,000	160	161.00	586.58
1,000	180	180.87	1007.29
1,000	200	198.88	1171.16
1,000	220	233.39	1308.92
1,000	240	243.32	1449.05
1,000	260	266.88	
1,000	280	279.45	
1,000	300	300.86	

Conclusions

Using a single subscriber queue we can see that the elapsed time taken to deliver messages is strictly proportional to the number of subscribers, as one would hope. Using an individual queue for each subscriber significantly increases the elapsed time. The chart below shows that when using multiple queues there was a sharp increase in elapsed time at around 160 subscribers. This could probably be postponed by increasing the qm.ini parameter "*OpenCacheSize*".

A general recommendation is to use shared reply queues with correlid during application design rather than having a dedicated queue for each application. Maintain the application/queue ratio such that the number of queues is less than "OpenCacheSize".

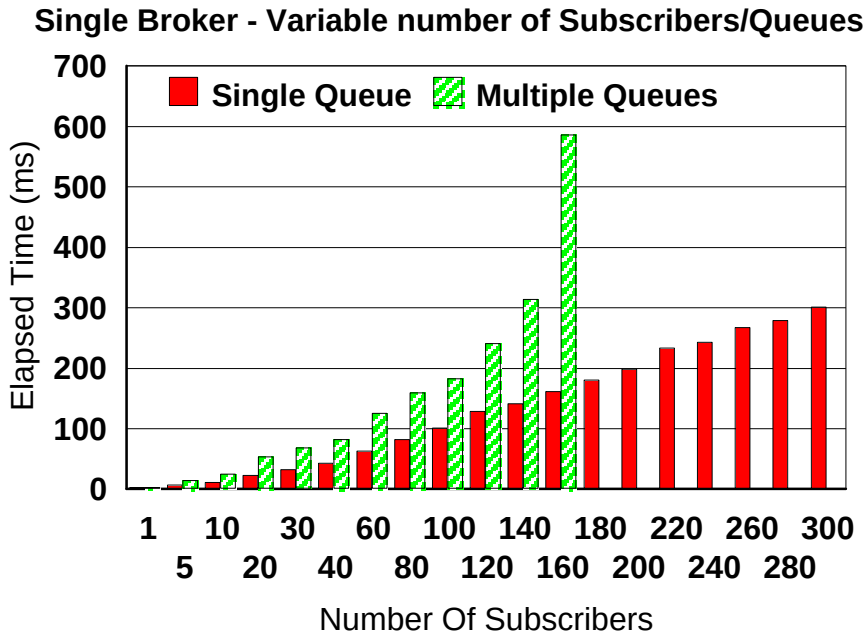


Chart 2: Total elapsed time (milliseconds) to deliver a publication to a set number of subscribers.

"Loopback test" with two brokers

This evaluation was designed to investigate performance with multiple (two) brokers. The evaluation consisted of three different types of test. The first type used the genuine broker code. The second used multiple applications to simulate the action of the brokers. The third test simply used queue definitions to measure round trip times between two queue managers.

By comparing results from the first and second tests we can isolate the cost of the broker activity.

Results from the third test give us the elapsed time to simply send messages through the channels.

The Genuine Broker test

Two brokers, B_1 and B_2 were set up with B_1 acting as the parent of B_2. Two Streams, S_1 and S_2 were set up on these brokers. An application published to broker B_1's stream S_1. The message was sent to broker B_2's stream S_1 in response to a subscription from a separate application running on B_1. This subscription specified a reply queue of broker B_2's S_2 publication stream.

The publication was sent back to broker B_1's S_2 stream in response to a subscription from the originating application running on B_1.

Simulated Broker test

An application running on queue manager B_1 PUT a message to stream queue S_1. This message was retrieved from the stream queue by an application acting as broker and MQPUT to a remote stream queue S_1 on queue manager B_2. At queue manager B_2 a second "broker" application retrieved the message on B_2's S_1 queue and placed it on queue S_2. The message was again processed by a simulated broker and forwarded to stream queue S_2 on B_1 before finally being placed on the reply queue at queue manger B_1 where it was retrieved by the originating application.

Round trip using "Queue Magic"

An application on queue manager B_1 PUT a message to a remote stream queue on queue manager B_2. The queue definition on B_2 resolved to the subscriber reply queue back on queue manager B_1.

Results

The table below shows average round trip times (in milliseconds) fro m the time of initial publication until arrival at the reply queue on queue manager B $_1$. The abbreviations in the table headings are as follows:

- **Broker**, **Simul**, **Chl** indicates the type of test - ge nuine broker, Simulated broker or round trip using queue definitions.
- **Persist** - persistent messages were used
- **NP,SP** - non persistent under syncpoint
- **SPIP** - non persistent with the *SyncPointIfPersistent* option enabled

Message Size	Batch Size	Persist Broker	Persist Simul	Persist Chl	NP,SP, Broker	NP,SP Simul	NP,SP, Chl	SPIP Broker	SPIP Simul	SPIP Chl
10	1	154.85	152.88	94.22	26.65	25.23	10.87	20.37	19.47	8.56
100	1	155.34	153.10	96.10	26.53	25.31	10.75	19.95	19.67	8.90
1,000	1	176.26	173.53	99.46	27.46	25.81	11.94	21.09	20.45	9.29
10,000	1	266.30	264.04		32.14	29.50	13.62	25.94	25.33	10.50
10	5	95.74	93.49	29.04	21.33	18.44	5.17	13.48	12.26	4.85
100	5	97.02	96.31	28.53	21.08	18.35	5.37	13.33	13.51	5.14
1,000	5	108.78	108.04	31.96	22.06	20.05	6.37	14.18	13.66	5.50
10,000	5	201.36	201.02		26.33	24.25	8.55	18.40	19.42	8.39
10	10	81.56	78.24	19.38	20.28	17.88	5.24	12.39	11.47	4.74
100	10	83.58	81.08	22.21	20.87	17.30	4.73	12.83	11.93	4.77
1,000	10	83.14	92.41	27.89	21.41	18.76	5.18	13.25	12.39	5.31
10,000	10	179.64	182.77		27.34	25.12	9.77	18.55	19.32	8.99
10	15	78.59	74.98	18.13	20.34	18.00	4.62	12.40	11.68	4.68
100	15	79.54	74.73	19.97	19.77	17.47	5.02	12.34	11.66	4.64
1,000	15	88.68	87.05	23.27	20.49	18.33	5.42	12.92	12.41	5.41
10,000	15	170.77	173.29		28.03	26.29	11.57	18.93	19.65	9.78
10	20	74.80	74.07	17.28	20.17	17.04	4.59	12.15	11.89	4.30
100	20	76.71	72.92	17.83	19.91	18.25	4.85	12.17	10.72	4.52
1,000	20	84.50	84.44	21.10	20.50	18.12	5.24	13.25	11.91	5.21
10,000	20	164.46	168.53		28.17	26.32	10.84	19.10	19.27	9.69
10	25	71.49	69.52	14.98	19.45	17.18	4.47	11.94	10.67	4.34
100	25	72.64	72.59	16.83	19.88	18.18	4.60	12.13	10.86	5.18
1,000	25	83.47	79.07	18.34	20.10	17.81	4.84	12.82	11.61	4.78
10,000	25	165.76	165.23		28.56	27.83	10.59	20.12	21.00	9.81
10	30	71.22	66.45	15.18	19.48	18.74	4.61	11.82	10.94	4.42
100	30	72.67	70.32	14.75	19.69	17.34	4.55	20.12	11.12	4.10
1,000	30	81.47	77.74	17.89	20.20	17.54	4.82	11.82	12.04	4.60
10,000	30	159.51	181.86		28.43	27.36	12.72	12.02	19.85	10.24

Conclusions

Persistent Messaging

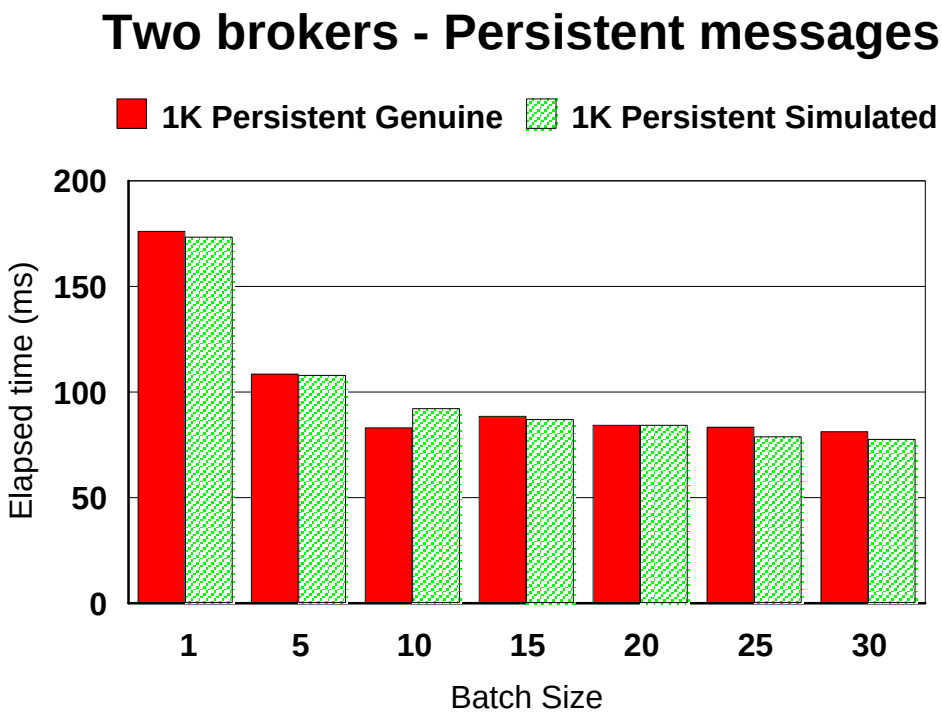
The results show that the overhead within the brokers is very small compared to the basic cost of message delivery for persistent messages (see chart below). There is a significant saving to be had by batching publications - up to a batch size of around 5. Beyond a batchsize of about 5 the savings are much less significant.

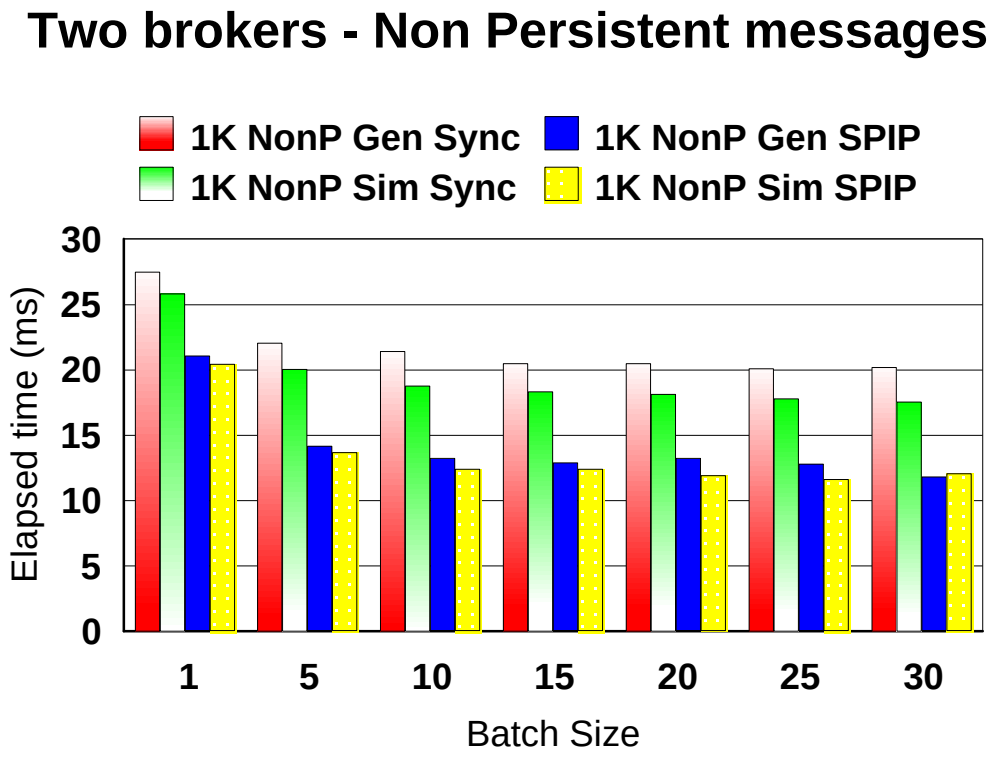
Non Persistent Messaging

These results show that the effect of batchsize to be less significant for non persistent messaging compared to persistent messaging. This is not surprising, syncpointing non persistent messages does not force a write to the log. Publishing with a batchsize greater than about 5 gives an insignificant benefit.

The overhead of the broker is more significant as a proportion of the total delivery time. The SyncPointIfPersistent option gives a very worthwhile saving when using two brokers.

Note more work needs to be done to evaluate the cost of scaling beyond two brokers.




Two brokers - Non Persistent messages
1K NonP Gen Sync
1K NonP Gen SPIP
1K NonP Sim Sync
1K NonP Sim SPIP
Elapsed time (ms)
0
5
10
15
20
25
30
1
5
10
15
20
25
30
Batch Size

Bank scenario

This evaluation was intended to simulate a configuration planned by a bank participating in the Publish/Subscribe ESP. It's purpose was to demonstrate such a configuration could achieve a message throughput rate sufficient to satisfy their requirements.

The test configuration consisted of five queue managers each running on a single RISC/6000 - J50.

A single application connected to the first queue manager published messages at a controlled rate.

These messages were forwarded to a second queue manager acting as a broker. The remaining three queue managers each had an application which registered subscriptions for the messages being published. The rate at which publications were issued was increased until a QDepthHi event was triggered on the broker's stream queue. This indicated that the broker was unable to process messages at a given rate. The test was repeated using persistent messages of various sizes.

The publisher and broker queue managers each had their log files on dedicated SSA disks. They also had dedicated disks for their queue files, although these were ordinary Fast/Wide SCSI.

Of the subscriber queue managers, one had it's log on a dedicated SSA disk, the remaining two shared an SSA disk. The subscribers all shared a common SCSI disk for their queue files.

The batchsize for all MQSeries sender channels was allowed to take the default (50).

Results

Message Size	Max throughput (msgs/sec)
10	49
100	45
1,000	39
3,000	33
6,000	24
10,000	20

Conclusions

A maximum throughput of 20 messages/second using 10K persistent messages was achieved which is believed to be twice the rate initially required by the bank. The limiting factor seemed to be the high utilisation of the disks containing the queue files for the both

the publisher and the broker. Utilisation on these disks exceeded 80% when publishing twenty 10K messages per second.

The RISC/6000 - J50 was not CPU constrained. In fact CPU utilisation was low.

Placing the queue files for both the publisher and broker on SSA disks would undoubtedly have improved performance. The disk layout is crucial in optimising persistent messaging. A general recommendation is to place the logs and queue files of every queue manager on dedicated SSA disks.