MQSeries Integrator for Sun Solaris – v1.1
Maximizing persistent message throughput

Edition 1.0

June 9, 2000

Brian P. Bell
IBM Global Services

Table of Contents

Notices

This report is intended to help understand the performance characteristics of MQSeries Integrator for Sun Solaris v1.1, and the scalability of the solution. The information is not intended as the specification of any programming interfaces that are provided by MQSeries or MQSeries Integrator.

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.

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 of the techniques is the responsibility of the reader. Much depends on the ability of the reader to evaluate these data and project the results to their operational environment.

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

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- DB/2
- IBM
- MQSeries
- MQSeries Integrator
- RS6000

The following terms, used in this publication, are trademarks of the Sun Corporation in the United States or other countries or both:

- Solaris

The following terms, used in this publication, are trademarks of the EMC Corporation in the United States or other countries or both:

- EMC

The following terms, used in this publication, are trademarks of the Veritas Corporation in the United States or other countries or both:

- Veritas

Acknowledgments

The following individuals contributed in varied ways to developing the presented information.

Satish Basvapatri
Jeff Gallup
Jim Lessard
Paul Moyer
John Spray
Rae Dean Walby
Corey Walker
Tom Walstad
Satish Venkatasubbu
(Alphabetically Listed)

Preface

This document outlines aspects of the MQSeries interface and MQSeries Integrator (MQSI) performance within the Solaris environment as described within. It is intended to help persons who are investigating IBM MQSeries and MQSI implementations for positioning them within their installation's needs.

NOTE: The information in this publication is supplemental to, and not intended as the replacement of, any product documentation. This information was developed in conjunction with use of the environment specified, and is limited in application to those specific hardware and software products and levels.

The Audience

This document is designed for people who:

- Want to explore the performance considerations of MQSI for Solaris V1.1 with MQSeries V5.1
- Will be performance testing MQSI for Solaris V1.1 in order to provide Technical Support
- Will be designing and developing implementations that use MQSI for Solaris V1.1. Users should have a general awareness of Solaris (UNIX), MQSeries, MQSI, and be familiar with the Persistent/Non-Persistent Message concept, as well as Unit of Work operation to get the best out of this document.

What is in this document

- A set of graphs showing the MQSI performance aspects in a variety of application configurations and implemented on various Solaris configurations
- Interpretation of these graphs, and the implications for application design

Introduction

Many companies are facing an environment where enterprise application integration has become increasingly important to their information technology operations. Additionally, the changing business environment requires a migration beyond Intra-Company integration, and often calls for Inter-Company integration. The use of MQSeries, MQSeries Integrator and MQSeries Workflow can bring many benefits to an intra-Company integration, as they enable asynchronous processing and provide cross-environment facilities. This document provides detailed information on the performance of the MQSeries Integrator Infrastructure as it transforms real-world transaction data within a Solaris 2.6/SunOS 5.6 environment.

Objective

The Objective of this testing was to determine the application configuration and tuning that would maximize the throughput for *Persistent MQSeries messages* using MQSeries and MQSeries Integrator Infrastructure.

*The initial hardware and software configurations were obtained from the IBM support pacs. The configuration was then altered to optimize performance while maintaining a secure production environment. The configuration and tuning of the hub was performed to achieve the best throughput performance in the installed environment (The settings in this test are considered optimal for this particular production environment. They **are not** the fastest possible. Assured message delivery has a cost).*

Test Environment

All of the tests were conducted on two pair of Solaris 2.6 (SunOS 5.6) machines with common system software stack as described in the Physical Environment section of this document. The first pair (P1 and P2) make up the P hub and the second pair (M1 and M2), make up he M hub.

The *preliminary* tests were conducted to determine the optimum application configuration for the fully configured test machine, where 'optimum' was defined as the greatest message per second throughput rate. This involved varying the number of:

- Queue Managers
- MQSeries Input Queues
- MQSI Rules Engine Daemons

Once the optimum application configuration was found, more tests were run to further configure the hardware.

Test Model

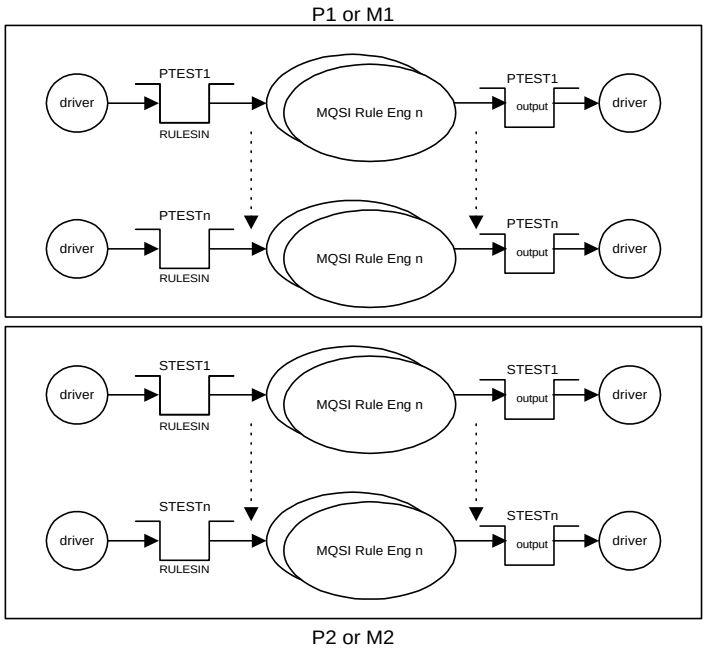

Figure 1

Figure 1 - Preliminary Test Model Diagram

Figure 1 illustrates the model that was used as a starting point for the testing. The driver application loaded and maintained the input queue depth of 100 messages to insure MQSI would process messages at the maximum rate. The driver program put messages into the RULESIN queue and processed messages from the output queue, recording data every 2000 messages until total throughput reached 40,000 messages. The data recorded was saved in a file for later analysis.

Each of the 40,000 messages put to the input queue were 1024 bytes in length. After rules engine processing, the messages were placed on the output queue. The reformatted messages place on the output queue were also 1024 bytes in length.

The test model was processed as follows:

- The driver programs were started and the respective input queue level loaded to 100 messages by the driver program.
- The MQSI rules engines were started on both systems and began processing messages from the input queue.
- Every time the rules engine processed another message the driver program put a persistent message to the input queue maintaining 100 messages in the input queue (a parameter specified the number of records per commit – Syncpoint control).
- One or more instances of the MQSI rule engine would read the input queue (RULESIN), with the resulting output being written to a single MQSeries queue (OUTPUT) per Queue Manager ([S or P]TESTn).
- The driver program would read the output queue under Syncpoint, and write the performance information to a report file.
- A commit was issued after the appropriate number of messages, as passed in a program parameter.
- All messages processed by the rules engine would undergo a single reformat and routed to one queue.

The following MQSI environment variables were set:

- LogLevel was set to 2 in the .mpf file to log only errors and fatal errors
- All queues on the hub queue manager are persisted: DEFPSIST(YES)
- NN_ALERT was not set therefore it defaults to ON

The settings of the above MQSI parameters were carefully considered and mirror the proposed settings in the production environment.

Hypothesis

Before testing began, it could have been assumed that:

- The more queue managers added, the greater the throughput
- Disk input/output would be a bottleneck
- Increasing the number of MQSI Rules Daemons per input queue would improve performance

Various vendor white papers and expert council supported the assumptions.

Physical Environment

P Hub

P1		
	Function	Production P Hub
	IP Address	
	Logical IP Address	Required for fail-over
	Host Name	P1
Hardware	Machine Type	SUN E4500
	Processors	4 x 400 MHz – 8Mb of external cache
	Storage	Presented as 6 11GB mirrored LUNS mirrored on the EMC side, stripped via Veritas Volume Manager on the server
Software/OS	OS Version	Solaris 2.6 (SunOS 5.6)
	MQSeries Version/PTF	MQSeries for Solaris 5.1/02
	MQSeries Manager Name	PTEST1, PTEST2, …,PTESTn
	MQSeries Cluster Name	PHUB
	MQSeries Integrator Version/PTF	MQSI V1.1 for Solaris and UDB
	Universal Database Version/PTF	DB2 V6.1
	Veritas File System	Version 3.2
	Database Name	PERFORM
	Machine Location	PHX IPC

P2		
	Function	Production P Hub
	IP Address	
	Logical IP Address	Required for fail-over
	Host Name	P2
Hardware	Machine Type	SUN E4500
	Processors	4 x 400 MHz – 8Mb of external cache
	Storage	Presented as 6 11GB mirrored LUNS mirrored on the EMC side, stripped via Veritas Volume Manager on the server
Software	OS Version	Solaris 2.6 (SunOS 5.6)
	MQSeries Version/PTF	MQSeries for Solaris 5.1/02
	MQSeries Manager Name	STEST1, STEST2, …,STESTn
	MQSeries Cluster Name	PHUB
	MQSeries Admin UserID	Mqm
	Universal Database Version/PTF	DB2 V6.1
	Database Name	PERFORM
	Veritas File System	Version 3.2

M Hub

M1		
	Function	Production M Hub
	IP Address	
	Logical IP Address	Required for fail-over
	Host Name	M1
Hardware	Machine Type	SUN E4500
	Processors	4 x 400 MHz – 8Mb of external cache
	Storage	Sun StorEdge A5200 presented as mirrored (7 x 9.1 GB, 10,000rpm low profile FC-AL drives) One of the disks is a hot spare
Software	OS Version	Solaris 2.6 (SunOS 5.6)
	MQSeries Version/PTF	MQSeries for Solaris 5.1/02
	MQSeries Manager Name	PTEST1, PTEST2, …,PTESTn
	MQSeries Cluster Name	MHUB
	MQSeries Integrator Version/PTF	MQSI V1.1 for Solaris and UDB
	Universal Database Version/PTF	DB2 V6.1
	Database Name	PERFORM
	Veritas File System	Version 3.2

M2		
	Function	Production M
	IP Address	
	Logical IP Address	Required for fail-over
	Host Name	M2
Hardware	Machine Type	SUN E4500
	Processors	4 x 400 MHz – 8Mb of external cache
	Storage	Sun StorEdge A5200 presented as mirrored (7 x 9.1 GB, 10,000rpm low profile FC-AL drives) One of the disks is a hot spare
Software	OS Version	Solaris 2.6 (SunOS 5.6)
	MQSeries Version/PTF	MQSeries for Solaris 5.1/02
	MQSeries Manager Name	STEST1, sTEST2, …,sTESTn
	MQSeries Cluster Name	MHUB
	MQSeries Integrator Version/PTF	MQSI V1.1 for Solaris and UDB
	Universal Database Version/PTF	DB2 V6.1
	Database Name	PERFORM
	Veritas File System	Version 3.2

Note: the only difference between the M hub and the P hub was the storage device used.

Initial Hub Disk Architecture – mirroring, striping, logging, etc (figure 2)
Each hub was set-up with a 24 disk array. Twelve of the disks were reserved for a mirror of the production disk environment. The mirrored disk would automatically take over in the case of a failure of any of the primary disks. Of the twelve primary disks, six were allocated for queue managers and logs on each system. The MQSeries queues were striped across three of the disks and the log files were striped across the other three disks for each Sun box. Configurations were then placed in Veritas software to allow the disk, queues, log files and subsystems to fail over between machines in the case of failure of one of the Sun boxes

The system was configured in this manner for a number of reasons.

- Every time a message was persisted to a queue, a write to disk occurs in two locations (the log file and queue). Separating log files and queue data allowed different disks to I/O at the same time therefore increasing performance by shortening total I/O time.
- Separating the queue data and logs also reduced the chance of loosing data. In the event of a queue disk failure, you would have the logs to recover. If the logs and queue data was not separated, there would be a chance of losing a log and its corresponding queue with the loss of one disk. This would lead to lost data and possible system crash.
- The mirror of production disk allowed for continuation of processing if a disk failed. In the event of a single or multiple disk failure, the mirror of the disk would take over the primary position until the failed disk was replaced.
- The fail-over of the complete queuing sub-system was devised to allow continued processing during a catastrophic Sun system failure.

The hub configuration was designed for high availability and performance with multiple redundancies to insure continuation of processing and to nearly eliminate the possibility of loosing data (See figure 2).

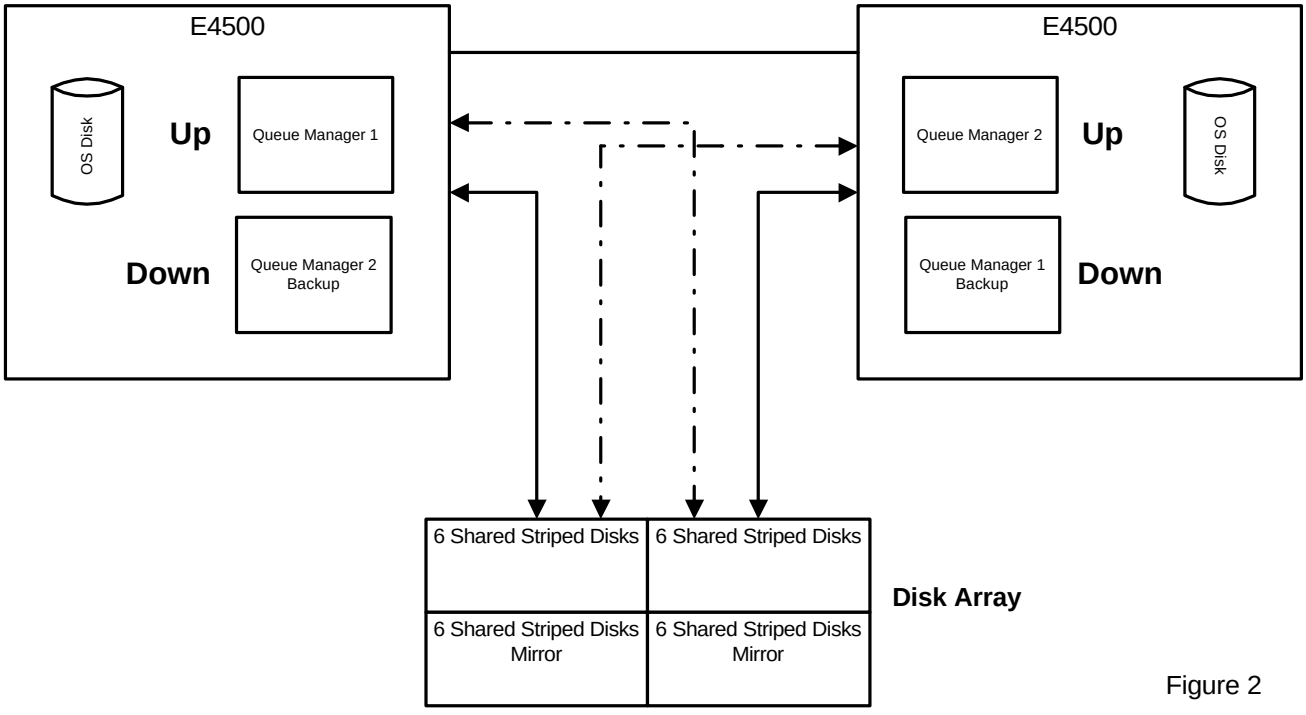

Figure 2

Final Testing Environments

The hardware and software environments were altered during the several of the tests to optimize performance. For the test results reported, the hardware environment was the final testing environment and considered optimal at this point in time. The software/middleware configuration changes were altered during tests to pin point optimal configuration. Refer to the Test Case Section to see the description of the software/middleware setup.

Test Cases

Several test cases were run to tune the parameters to insure maximum throughput. These test cases included varying the rules and formats. The formats were all flat, multiple field formats. After several tests were conducted it was surmised that the reformat of data from ASCII to EBCDIC produced the lowest throughput of any flat reformat. The EBCDIC test was used as a baseline for all tests.

Test cases one, two and five were performed on the P hub and test cases three and four were performed on the M hub. It was necessary to perform tuning on both hubs because the hubs used different types of disk. Lessons learned in each test case were applied to the next test case.

Preliminary Test Cases

Test Case 1: Determine maximum number of rules engines (P Hub)
Description: Tests were run to determine optimal the number of rules engines for one queue manager. The test was set up to run 1, 2, 3, 4 and 8 rules engines against one input queue. This test was first run on the internal disk and then on the striped disk, described in Physical Environment section.

Results: The results (figure 3) in both striped and non-striped disk tests, performance peaked most frequently during the 2 rules engines test when running against 1 queue. The message throughput nearly tripled when the test was run on the striped disk. The gain was attributed to two characteristics. In the non-striped test, both queue data and log files were written to the Sun internal disk. In the striped test, the queue data and the log files each were configured to stripe across three high RPM disk. Separation of workload and high performance disks resulted in greater throughput.

The rules engine processes consumed the greatest amount of CPU, 10-15% per rules engine. MQ processes consumed the majority of the remainder of the CPU and each of the driver programs consumed less than 1% of the CPU.

Results from tests performed on P2

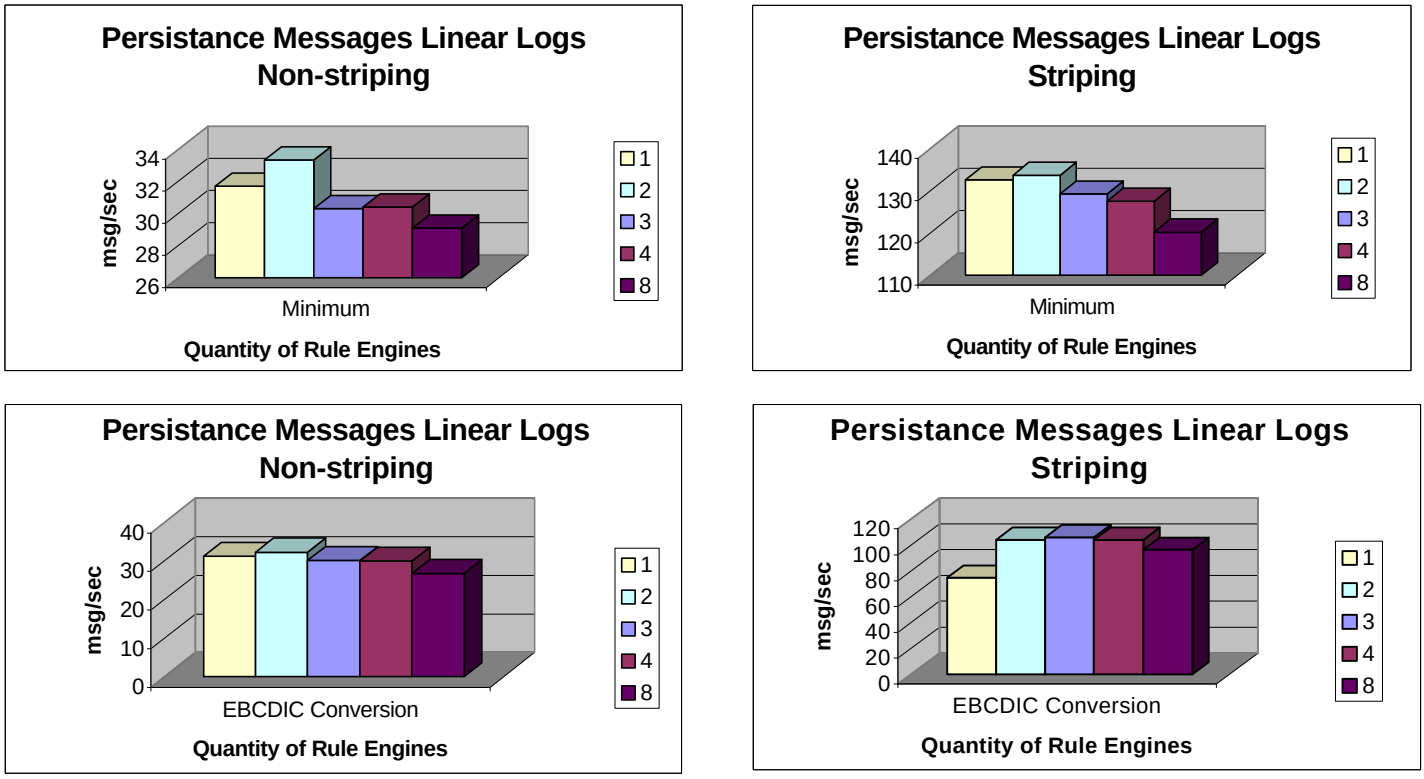

Figure 3

Test Case 2: Multiple queue managers with multiple rules engines (P Hub)
Description: Test 2 was similar to the striped test in test case 1. Test case 2 ran 1 and 2 rules engines on with multiple queue managers.

Test case 2 represented the volume for our final objective. Two basic concepts were explored:

- Expand the breadth of the system through additional Queue Managers, and therefore, additional Queues and MQSI Rule Engines
- Compare normal hub operation verses fail-over operations. Benchmark the total hub throughput during normal and failed-over operation.

Results: The best performance was offered by the configuration highlighted in the charts (fig. 4,5,6). During normal hub operations, the optimal performance configuration was 5 queue managers and one rule engine per queue manager (figure 4). The run achieved an average of 220 messages per second through a physical box and 440 messages per second through the logical hub.

The same test was run during the fail-over operation. During this event, the optimal configuration for performance was 4 queue managers and one rule engine per queue manager. The fail-over performance test was conducted twice. One test was completed for P2 failed-over to the P1 (figure 5) and one for P1 (figure 6) failed-over to P2. Each of the runs achieved 280 messages per second through the physical and logical hub.

It became evident that there was a practical limit to the scalability of this solution with the current hardware configuration. As more queue managers and rules engines were added, CPU idle time approached 0 and throughput was maximized. Test Case 2 demonstrated contrary to our initial thought, that CPU was the limiting throughput resource.

In each of the test cases, the scenario was executed repetitively, and data captured by *vmstat* and *iostat* was used to determine if the scenario had exhausted any system resource (see figure 4, 5 & 6).

Previously conducted testing indicated an increase in Messages/Second for a given test scenario, when the number of MQSI rule engines was increased from one to two per input queue and when adding an additional queue manager. In test case number 2, we did not experience the same performance gain by adding another rule engine. The degradation of performance was attributed to contention for CPU.

Normal Hub Operation Statistics												
P1						Total Hub Throughput	P2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait		# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait
1	1	86	86	70	20	188	1	1	89	89	69	15
1	2	66	132	50	23	270	1	2	69	138	49	20
1	3	55	165	33	21	342	1	3	59	177	30	17
1	4	49	196	21	16	404	1	4	52	208	18	12
1	5	43	215	12	10	440	1	5	45	225	8	6
2	1	112	112	53	23	232	2	1	120	120	53	23
2	2	84	168	24	18	350	2	2	91	182	22	15
2	3	61	183	9	8	387	2	3	68	204	7	6
2	4	48	192	3	2	396	2	4	51	204	2	1
2	5	39	195	1	0	400	2	5	41	205	1	0

Figure 4

Failed-over entire hub to P1 P1 Fail-over statistics					
P1					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Hub Msg/Sec	CPU Idle	IO Wait
1	2	68	136	47	19
1	4	59	236	30	16
1	6	46	276	7	5
1	8	35	280	1	0
1	10	27	270	1	0
2	2	92	184	20	13
2	4	67	268	4	3
2	6	41	246	1	0
2	8	30	240	1	0
2	10	24	240	1	0

Figure 5

Failed-over entire Hub to P2 P2 Fail-over statistics					
P2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Hub Msg/Sec	CPU Idle	IO Wait
1	2	68	136	47	18
1	4	59	236	30	16
1	6	46	276	7	5
1	8	35	280	1	0
1	10	27	270	1	0
2	2	93	186	20	13
2	4	68	272	5	4
2	6	41	246	1	0
2	8	30	240	1	0
2	10	24	240	1	0

Figure 6

Test Case 3: Multiple queue managers with multiple rules engines (M Hub)

Description: Test 3 was identical to test case 2 except the test was run on the M hub.

Test case 3 represented a comparison between the two hubs. The type of storage device was different on each of the hub. The P hub's storage device included built in cache, which allowed for higher performance. It was assumed there would be a performance variance between the two hubs due to the storage device.

Results: The best performance was offered by the configuration highlighted in chart (fig. 7). During normal hub operations, the optimal performance configuration was 5 queue managers and one rules engine per queue manager (figure 7). The run achieved an average of 127.5 messages per second through a physical box and 255 messages per second through the logical hub.

Previously conducted testing on the P hub yielded a considerably higher throughput. The identical test in P resulted in 440 messages per second compared to 255 on the M hub. The throughput in M was 42% less than the throughput in P. Figure 8 depicts the difference between the two hubs as more queue managers are added.

Due to the low throughput of the normal operation of the configuration, the fail-over test was not conducted. It was concluded that further tuning would be required to reach a higher level of throughput.

Normal M Hub Operation Statistics – Test Case 3												
M1						Total Hub Throughput	M2					
#Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait		#Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait
1	1	63	63	80	66	125	1	1	62	62	80	65
1	2	49	98	64	61	196	1	2	49	98	63	60
1	3	40	120	51	49	237	1	3	39	117	51	49
1	4	31	124	42	40	252	1	4	32	128	41	39
1	5	25	125	35	34	255	1	5	26	130	37	36
2	1	61	61	74	63	121	2	1	60	60	75	63
2	2	46	92	56	54	182	2	2	45	90	57	55
2	3	38	114	39	38	225	2	3	37	111	41	39
2	4	28	112	33	32	228	2	4	29	116	31	30
2	5	21	105	24	22	225	2	5	24	120	24	22

Figure 7

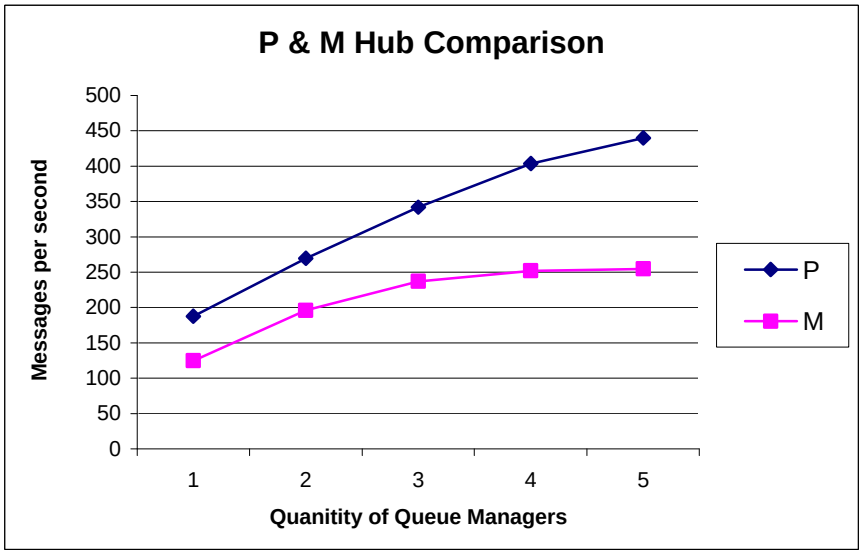

Figure 8

Test Case 4: One disk allocated per Queue manager and log (M hub)

Description: After several tests were run on the M hub, the optimal disk and file system allocation configuration was achieved.

Each queue manager and log file system was assigned to one physical disk. There were 6 production file systems each with their own disk. Figure 9 is a representation of the configuration. Each box represents a physical disk. The 6th file system remained idle for test case 4. It was designed for the production cluster repository queue manager. Since disk I/O was the limiting factor and 5 queue managers was determined to be optimal for maximizing the CPU, adding an additional queue manager would have decreased performance.

Disk 1 QM 1 Log 1	Disk 2 QM 2 Log 2	Disk 3 QM 3 Log 3	Disk 4 QM 4 Log 4	Disk 5 QM 5 Log 5	Disk 6 unused

Results: Under the new configuration performance increased considerably. During normal hub operations, the optimal performance configuration was 5 queue managers and one rules engine per queue manager physical box (figure 9). Total hub throughput averaged 375 messages per second through the logical hub. On fail-over, the hub's throughput decreased to 250 messages per second (figure 10).

The results from the test were satisfactory. The configuration from test case 4 was concluded to be the optimal hardware, software and operating system configuration for the M boxes.

Normal Hub Operation Statistics – Test Case 4												
M1						Total Hub Throughput	M2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait		# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait
1	1	55	55	80	61	113	1	1	58	58	77	53
1	2	45	90	65	58	184	1	2	47	94	64	54
1	3	41	123	52	48	249	1	3	42	126	49	45
1	4	40	160	36	35	316	1	4	39	156	36	34
1	5	37	185	24	23	375	1	5	38	190	24	23
2	1	56	56	69	56	110	2	1	54	54	69	57
2	2	51	102	45	43	202	2	2	50	100	44	41
2	3	49	147	23	22	291	2	3	48	144	24	22
2	4	43	172	10	9	344	2	4	43	172	9	8
2	5	37	185	4	3	370	2	5	37	185	4	3

Figure 9

Failed-over entire Hub to M2 M2 Fail-over statistics					
P2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Hub Msg/Sec	CPU Idle	IO Wait
1	2	47	94	64	55
1	4	40	160	36	34
1	6	36	216	14	13
1	8	30	240	4	3
1	10	25	250	1	0
2	2	56	112	43	40
2	4	44	176	10	8
2	6	33	198	2	0
2	8	25	200	1	0
2	10	20	200	1	0

Figure 10

Test Case 5: One disk allocated per Queue manager and log (P hub)
Description: The P hub was configured to be identical to the M hub configuration in test case 4.

Results: There was a slight increase in performance (40 messages/second) during regular hub operations and a slight decrease in performance (24-30 msgs/sec) during fail-over when test case 5 results were compared to the previous P test (test case 2). This configuration was concluded to be the final production configuration for both hubs.

Normal Hub Operation Statistics												
P1						Total Hub Throughput	P2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait		# Rule engines	# qmgrs	Avg Msg/Sec	Total Msg/Sec	CPU Idle	IO Wait
1	1	86	86	70	18	173	1	1	87	87	71	18
1	2	68	136	50	21	274	1	2	69	138	50	21
1	3	59	177	32	18	354	1	3	59	177	33	19
1	4	52	208	18	13	416	1	4	52	208	19	12
1	5	46	230	9	6	460	1	5	46	230	10	7
2	1	119	119	53	21	244	2	1	125	125	53	21
2	2	91	182	22	15	374	2	2	96	192	21	14
2	3	69	207	7	5	429	2	3	74	222	7	5
2	4	53	212	2	1	444	2	4	58	232	2	1
2	5	42	210	1	0	440	2	5	46	230	1	0

Figure 11

Failed-over entire Hub to P1 P1 Fail-over statistics					
P1					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Hub Msg/Sec	CPU Idle	IO Wait
1	2	70	140	48	18
1	4	53	212	17	10
1	6	41	246	3	1
1	8	32	256	1	0
1	10	25	250	1	0
2	2	99	198	19	11
2	4	59	236	2	0
2	6	38	228	1	0
2	8	28	224	1	0
2	10	23	230	1	0

Figure 12

Failed-over entire Hub to P2 P2 Fail-over statistics					
P2					
# Rule engines	# qmgrs	Avg Msg/Sec	Total Hub Msg/Sec	CPU Idle	IO Wait
1	2	69	138	49	17
1	4	54	216	16	9
1	6	41	246	3	1
1	8	32	256	1	0
1	10	25	250	1	0
2	2	101	202	18	11
2	4	62	248	1	0
2	6	41	246	1	0
2	8	30	240	1	0
2	10	24	240	1	0

Figure 13

Was Our Hypothesis True?

Interestingly, the outcome of this test was that we received the best performance with a configuration of 5 queue managers each with one rules engine per input queue during normal hub operations. The hypothesis was partly true *for this hardware configuration.* The largest surprise in the results was the discovery of the CPU not disk I/O being the limiting factor for performance. The one rule engine for optimal performance can be directly attributed to the contention for CPU.

Conclusions

The following conclusions were gathered from the tests.

CPU was the limiting resource in the hardware configuration. By adding more CPUs, performance could increase to a point until another resource, most likely disk I/O, would become the limiting resource. The current solution has the ability to scale from 4 processors per box to 14 processors per box.

Considerations

The following considerations should be noted:
The system performance could be improved with further tuning the environment. The following variations to this test could result in performance gains:

- Running the tests under various Syncpoint commits will produce varied results (Test were run with a Syncpoint of 25)
- Varying the LogBufferPages setting could produce different results (Tests were run with a setting of 17).
- A test for multiple input and output queues per queue manager was not tested and should be considered for further tests.

Finally

We believe additional performance gains can be achieved by adding additional hardware. The numbers in this report should not be viewed as a capacity ceiling. Instead they should be viewed as an indicator of the solutions performance potential of the current logical hub architecture. Additionally, this report provides some insight into the affect of the solution configuration, and the affect of the identified application tuning parameters.

More importantly, the hub architecture in both M hub and P hub was proven to provide enough throughput to exceed current needs.