IBM MessageSight Appliance v1.0 Performance Benchmarks

Tested by: IBM

Test date: April 22, 2013

Key Results

- The IBM MessageSight appliance is capable of storing over 400K messages per second to non-volatile media, with a power efficiency of over 1550 messages per Watt.
- 1M MQTT v3.1 devices can connect to the IBM MessageSight appliance in less than 60 seconds, with an average connection latency of less than 270 microseconds.
- The IBM MessageSight appliance can process messages from over 350K MQTT v3.1 applications each sending 1 message per second.
- The IBM MessageSight appliance can publish over 15M messages per second to large numbers of consumer applications.
- The IBM MessageSight appliance can deliver messages to latency critical applications with an average latency of 85 microseconds at a rate of 10K messages per second, and while the appliance was handling heavy loads.
- The IBM MessageSight appliance can forward over 200K MQTT QoS 0 messages per second to a WMQ backend.

Disclaimers

The evaluations described in this document were conducted under controlled laboratory conditions. Obtaining repeatable, measurable performance results requires a controlled environment with specific hardware, software, network, and configuration in an isolated system. Adjusting any single element may yield different results. Additionally, test results at the component level may not be indicative of system level performance, or vice versa. Each organization has its own unique requirements, and therefore may find this information insufficient for its specific needs.

Customers interested in a custom analysis for their environment are encouraged to contact IBM

Trademarks and Service marks

TABLE OF CONTENTS

Benchmark specifications

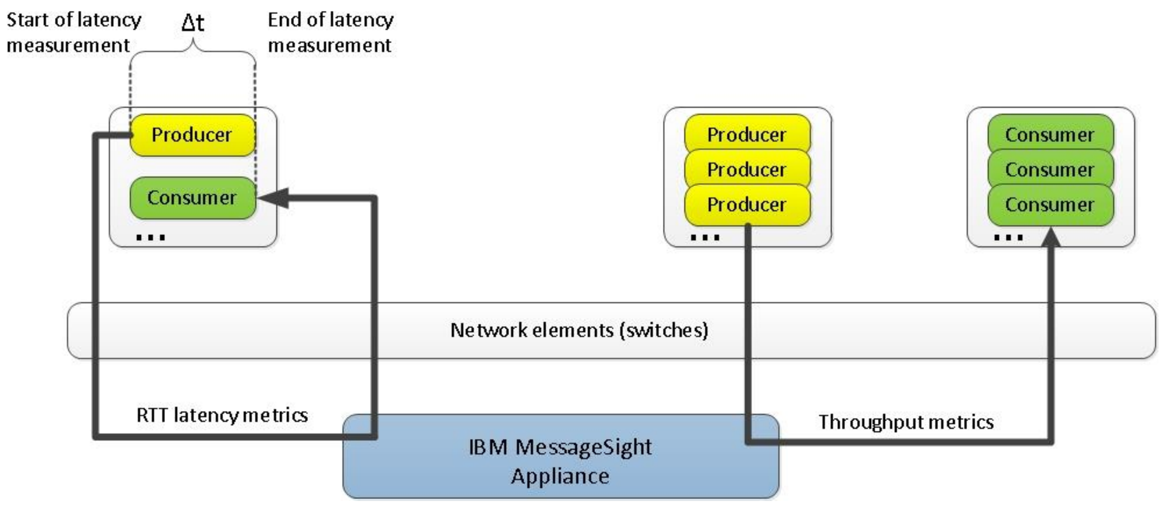
Overview

The performance benchmarks in this report test the ability of the IBM MessageSight appliance to handle high volumes of messaging traffic in a variety of configurations. The configurations tested in these benchmarks reflect application architectures and topologies which are commonly found in large scale networks of mobile and telemetry devices. The key performance metrics highlighted in this report are throughput, application to application latency, and data-center efficiency.

The hardware required to execute these benchmarks is described in detail in the System under test section of this report. A small number of machines are required to host the test harness programs which generate the messaging workloads which are processed by the IBM MessageSight appliance. The test harness programs (*mqttbench, JMSBenchTest, and mqbench)* used to generate messaging workloads, are described in detail in the Test harness section of this report. These benchmarks do not require any specialized hardware for time synchronization or wire capture; latency and throughput measurements are taken in the test harness programs.

The test harness programs used simulate producer and consumer applications were developed by IBM. Tests were conducted by IBM in controlled laboratory conditions.

Latency is reported as the round trip time from producer application to consumer application via the IBM MessageSight appliance. The clock source used for latency measurements is the TSC clock of one of the host systems used in the test. This requires the producer and consumer applications threads referencing the TSC clock to be collocated on the same CPU socket. Figure 1 below shows the general benchmark topology.


Start of latency measurement
Δt
End of latency measurement
Producer
Consumer
Producer
Producer
Producer
Consumer
Consumer
Consumer
Network elements (switches)
RTT latency metrics
Throughput metrics
IBM MessageSight Appliance

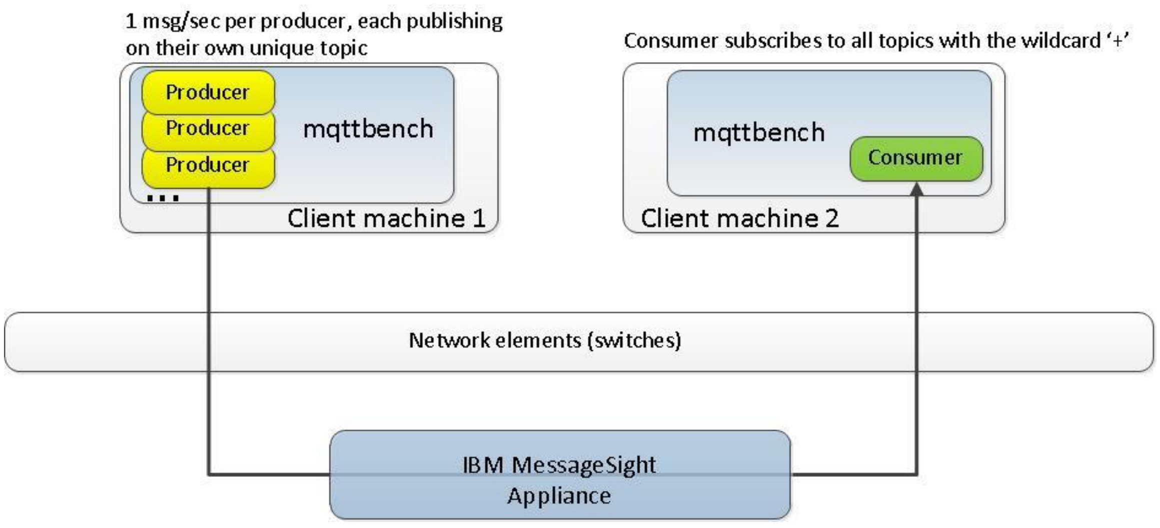
Test cases descriptions

This section describes the set of test cases and workloads used to generate the benchmark results included in this report. The results for each test case can be found by test case name in the Score card and Additional test results data sections.

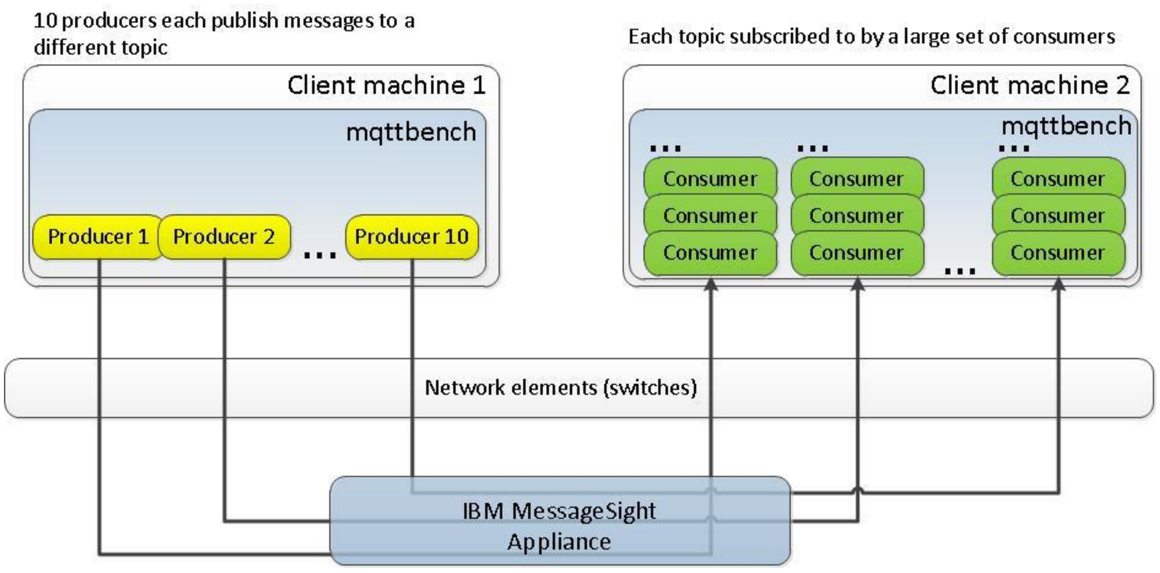
The messages used in the majority of the test cases described below are small messages, less than 512 bytes. The message sizes used in these test cases are intended to be representative of application data, such as GPS coordinates, sensor data, and other compact data that may be sent and received from mobile and telemetry devices. The LARGEMSG test case measures throughput for large messages.

FANIN

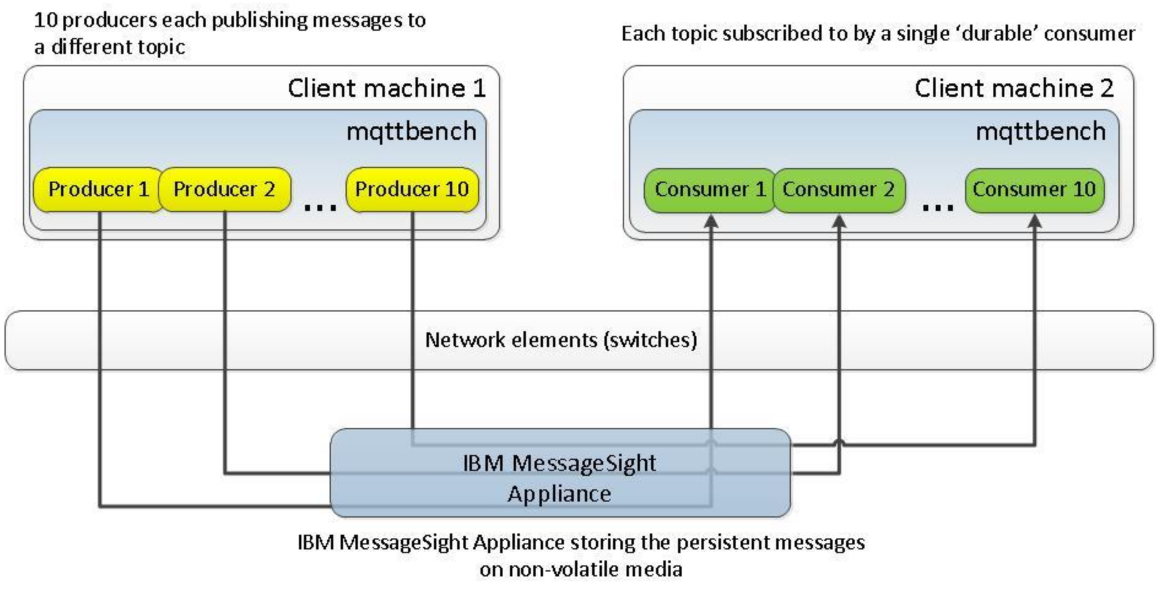
This test case measures the ability of the IBM MessageSight appliance to handle a large number of MQTT v3.1 producers publishing at a rate of 1 message/second per producer. Each MQTT (QoS 0) producer publishes on its own unique topic. A single MQTT (QoS 0) consumer subscribes to all the topics with a single wildcard subscription. On one client machine a single instance of *mqttbench* is started and is ready to consume the messages, subsequently a second instance of *mqttbench* on another client machine is started and begins generating the producer load.


1 msg/sec per producer, each publishing on their own unique topic
Producer
Producer
Producer
...
mqttbench
Client machine 1
Consumer subscribes to all topics with the wildcard '+'
mqttbench
Consumer
Client machine 2
Network elements (switches)
IBM MessageSight Appliance

FANOUT
This test case measures the ability of the IBM MessageSight appliance to publish messages to a set of topics with a large number of MQTT v3.1 subscribers. Ingress and egress message rates in to and out from the appliance are measured. In this test case 10 MQTT (QoS 0) producers each publish messages to a different topic. A large number of MQTT (QoS 0) consumers are subscribed to each topic. On one client machine a single instance of *mqttbench* starts and is ready to consume messages, subsequently a second instance of *mqttbench* on another client machine is started and begins generating the producer load.

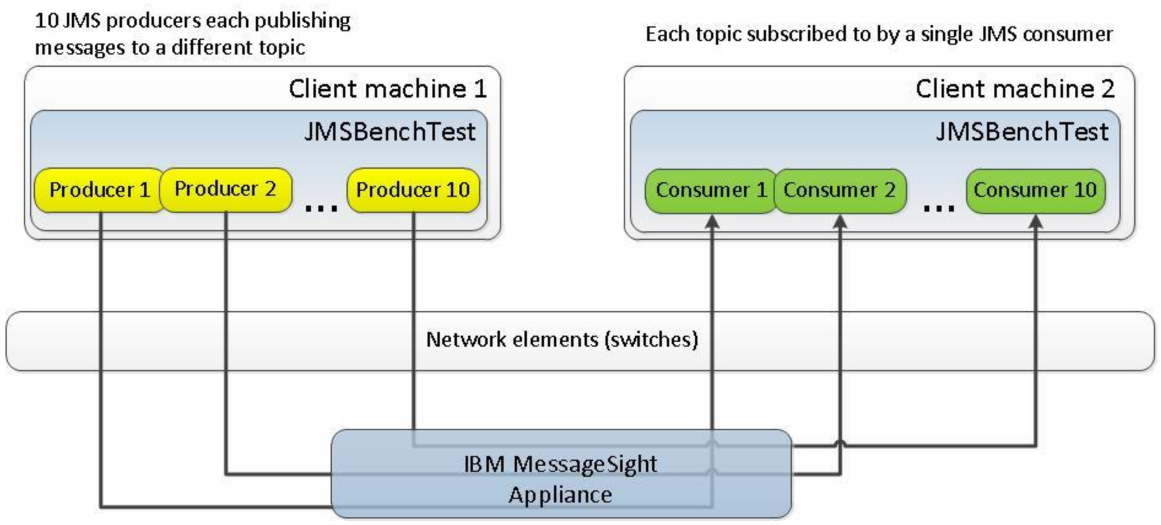

10 producers each publish messages to a different topic
Client machine 1
mqttbench
Producer 1
Producer 2
...
Producer 10
Each topic subscribed to by a large set of consumers
Client machine 2
mqttbench
Consumer
Network elements (switches)
IBM MessageSight Appliance

PERSISTENT
This test case measures the maximum persistent messaging (including storing messages to non-volatile media) rate which can be handled by the IBM MessageSight appliance. In this test case 10 pairs of MQTT v3.1 producers and consumers are sending and receiving persistent messages. Each pair of producers and consumers are publishing to and consuming messages from a different topic. On one client machine a single instance of *mqttbench* starts 10 durable MQTT (QoS 1) consumers and is ready to consume messages, subsequently a second instance of *mqttbench* on another client machine starts 10 MQTT (QoS 1) producers and begins generating the producer load. In addition to throughput, power consumption of the IBM MessageSight appliance is also measured.

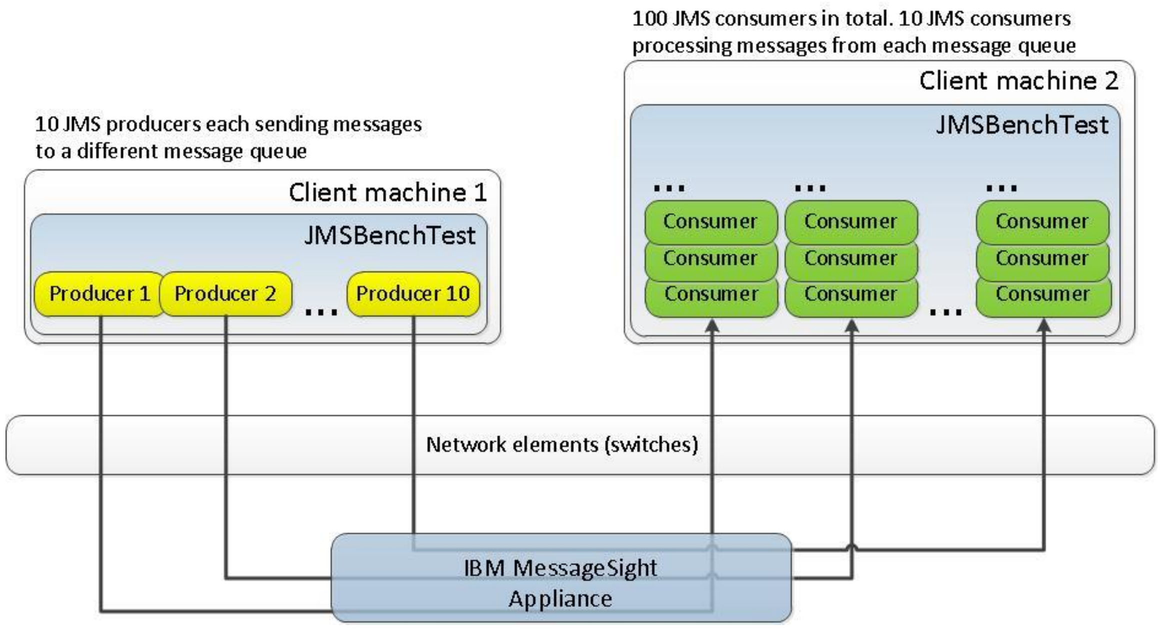

10 producers each publishing messages to a different topic
Each topic subscribed to by a single 'durable' consumer
Client machine 1
mqttbench
Producer 1
Producer 2
...
Producer 10
Client machine 2
mqttbench
Consumer 1
Consumer 2
...
Consumer 10
Network elements (switches)
IBM MessageSight Appliance
IBM MessageSight Appliance storing the persistent messages on non-volatile media

JMSPUBSUB

This test case measures the maximum throughput rate at which the IBM MessageSight appliance can handle JMS messages, published from producers and sent to consumers which use the IBM MessageSight JMS client library. In this test case, the lowest quality of service messaging ("fire-and-forget") allowed by the IBM MessageSight JMS client library is used. Fire-and-forget messaging is equivalent to MQTT QoS 0: non-persistent, non-transactional, and ACKing is disabled. This test leverages the IBM MessageSight JMS client library message cache. In this test case a single JVM is started on client machine 2. From this JVM, 10 *JMSBenchTest* consumer threads are started, each one creating its own JMS connection and session. Each consumer thread subscribes to a different topic. On client machine 1 a single JVM is started. From this JVM, 10 *JMSBenchTest* producer threads are started, each one creating its own JMS connection and session. Each producer thread publishes to a different topic.

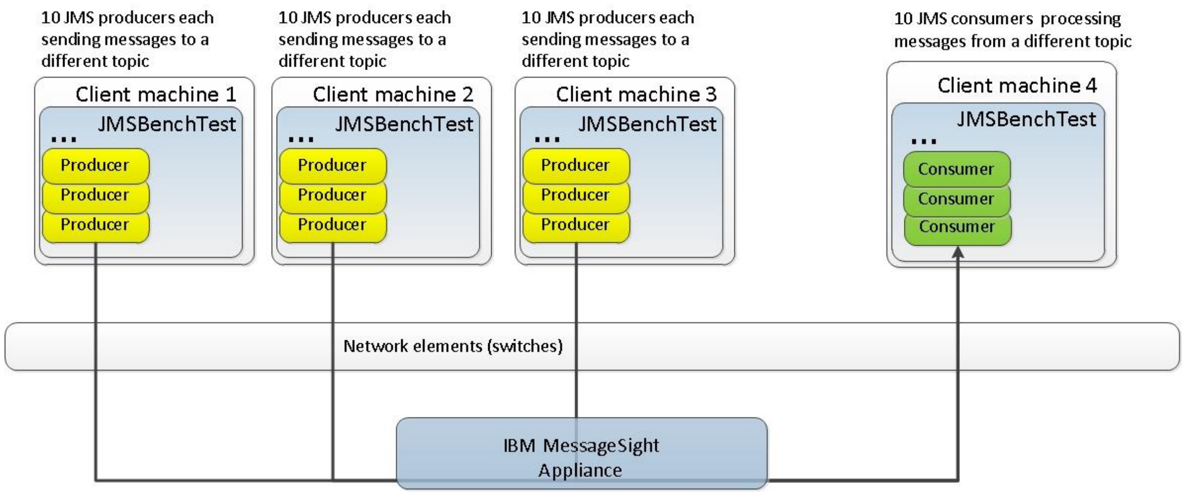

10 JMS producers each publishing messages to a different topic
Each topic subscribed to by a single JMS consumer
Client machine 1
JMSBenchTest
Producer 1
Producer 2
...
Producer 10
Client machine 2
JMSBenchTest
Consumer 1
Consumer 2
...
Consumer 10
Network elements (switches)
IBM MessageSight Appliance

JMSQUEUE
This test case measures the maximum throughput rate at which the IBM MessageSight appliance can deliver JMS messages to multi-consumer message queues. In this test case a single JVM is started on client machine 2. From this JVM, 10 *JMSBenchTest* consumer threads are started, each one creating 10 JMS connections, for a total of 100 JMS consumers. From each JMS connection a JMS transacted session is created. All consumers on each application thread receive messages from the same message queue. Consumers from different application threads receive messages from different message queues. The transaction size used by consumers in this test is 100 messages/transaction. On client machine 1 a single JVM is started. From this JVM, 10 *JMSBenchTest* producer threads are started, each one creating its own JMS connection and transacted session. Each producer thread publishes to a different message queue. The transaction size used by producers in this test is 100 messages/transaction. This test leverages the IBM MessageSight JMS client library message cache.

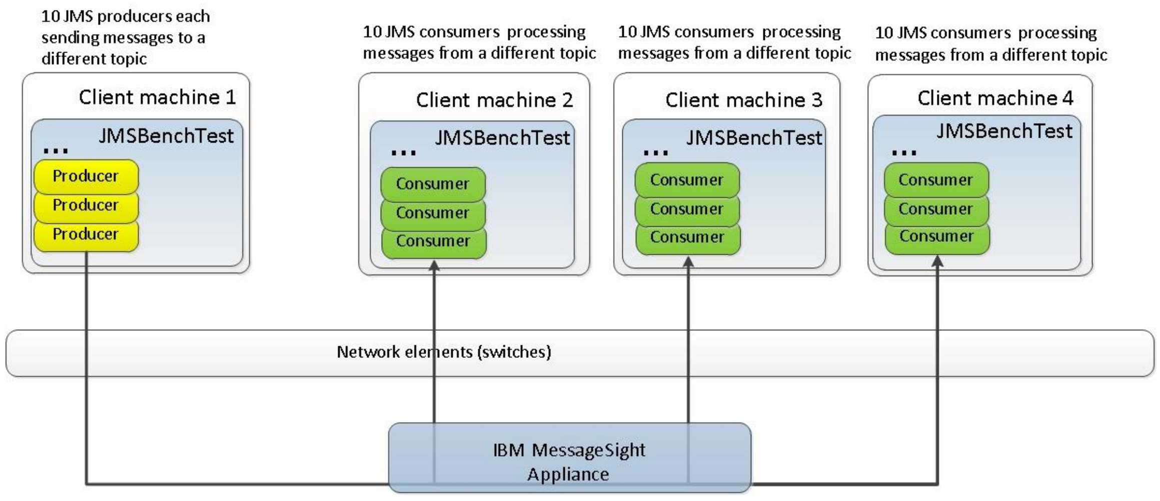

100 JMS consumers in total. 10 JMS consumers processing messages from each message queue
Client machine 2
JMSBenchTest
Consumer
10 JMS producers each sending messages to a different message queue
Client machine 1
JMSBenchTest
Producer 1
Producer 2
Producer 10
Network elements (switches)
IBM MessageSight Appliance

JMSPERSISTENT

This test case measures the maximum persistent messaging (including storing messages to non-volatile media) rate which can be handled by the IBM MessageSight appliance, when using JMS applications to generate the workload. In this test case, persistent messages are equivalent to MQTT QoS 1. The ACK mode used by the JMS consumers is client application ACK mode and the consumer applications send ACKs at a rate of 100 received messages per ACK. In this test case a single JVM is started on client machine 4. From this JVM, 10 *JMSBenchTest* consumer threads are started, each one creating its own JMS connection, non-transacted session, and message consumer. Each consumer creates a durable subscription to a different topic. On client machine 1, 2 and 3, a single JVM is started on each machine. From each JVM, 10 *JMSBenchTest* producer threads are started, each one creating its own JMS connection, non-transacted session, and message producer. Each producer publishes to a different topic, but the producers on different machines publish to the same 10 topics, thus 3 producers per topic and one consumer per topic.

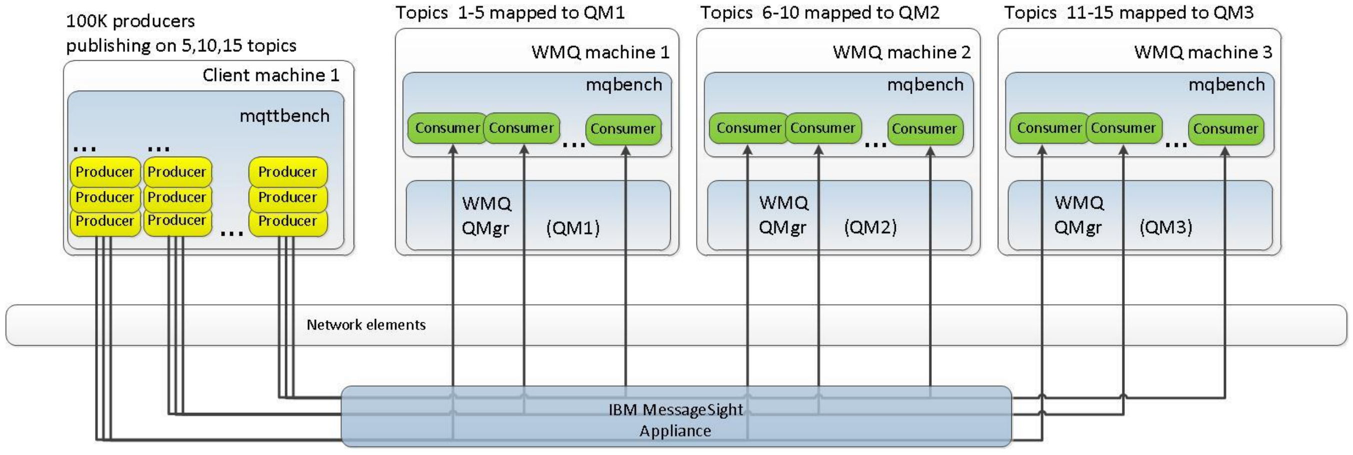

10 JMS producers each sending messages to a different topic
Client machine 1
JMSBenchTest
...
Producer
Producer
Producer
10 JMS producers each sending messages to a different topic
Client machine 2
JMSBenchTest
...
Producer
Producer
Producer
10 JMS producers each sending messages to a different topic
Client machine 3
JMSBenchTest
...
Producer
Producer
Producer
10 JMS consumers processing messages from a different topic
Client machine 4
JMSBenchTest
...
Consumer
Consumer
Consumer
Network elements (switches)
IBM MessageSight Appliance

JMSFANOUT
This test case measures the maximum throughput rate at which the IBM MessageSight appliance can handle JMS messages, published from producers and sent to consumers which use the IBM MessageSight JMS client library. In this test case, the lowest quality of service messaging ("fire-and-forget") allowed by the IBM MessageSight JMS client library is used. Fire-and-forget messaging is equivalent to MQTT QoS 0: non-persistent, non-transactional, and ACKing is disabled. This test leverages the IBM MessageSight JMS client library message cache. In this test case a single JVM is started on client machine 1. From this JVM, 10 *JMSBenchTest* producer threads are started, each one creating its own JMS connection, non-transacted session, and message producer. Each producer publishes to a different topic. On client machine 2, 3 and 4, a single JVM is started on each machine. From each JVM, 10 *JMSBenchTest* consumer threads are started, each one creating 10 JMS connections, non-transacted sessions, and message consumers. The ratio of producers to topics is 1:1 and the ratio of consumers to topics is 30:1, thus a fanout ratio of 1:30.


10 JMS producers each sending messages to a different topic
10 JMS consumers processing messages from a different topic
10 JMS consumers processing messages from a different topic
10 JMS consumers processing messages from a different topic
Client machine 1
JMSBenchTest
Producer
Producer
Producer
Client machine 2
JMSBenchTest
Consumer
Consumer
Consumer
Client machine 3
JMSBenchTest
Consumer
Consumer
Consumer
Client machine 4
JMSBenchTest
Consumer
Consumer
Consumer
Network elements (switches)
IBM MessageSight Appliance

WMQPUBSUB
This test case measures the maximum throughput rate at which the IBM MessageSight appliance can forward MQTT QoS 0 messages to a WebSphere MQ (WMQ) backend. The workload, in this test case, is generated from 100K MQTT QoS 0 producers. The workload is evenly distributed across a number of topics, which are mapped, using administratively created destination mapping rules in the appliance, to a number of WMQ queue managers. Thus, the workload is evenly distributed across multiple WMQ queue managers. The destination mapping rules used in this test case are of type, "appliance topic to MQ topic". In this test case a single WMQ queue manager is started on WMQ machine 1. Subsequently, a single instance of *mqbench* is started on WMQ machine 1 and subscribes to 5 MQ topics. The entire workload, from all 100K MQTT producers, is forwarded to these 5 MQ topics. On client machine 1, a single instance of *mqttbench* starts 100K MQTT QoS 0 producers and generates the workload, which is distributed across 5 MQTT topics. The destination mapping rules map each of the incoming 5 MQTT topics to a different outgoing MQ topic. The test is then repeated, but with additional WMQ queue managers. The load is distributing evenly across all the WMQ queue managers.

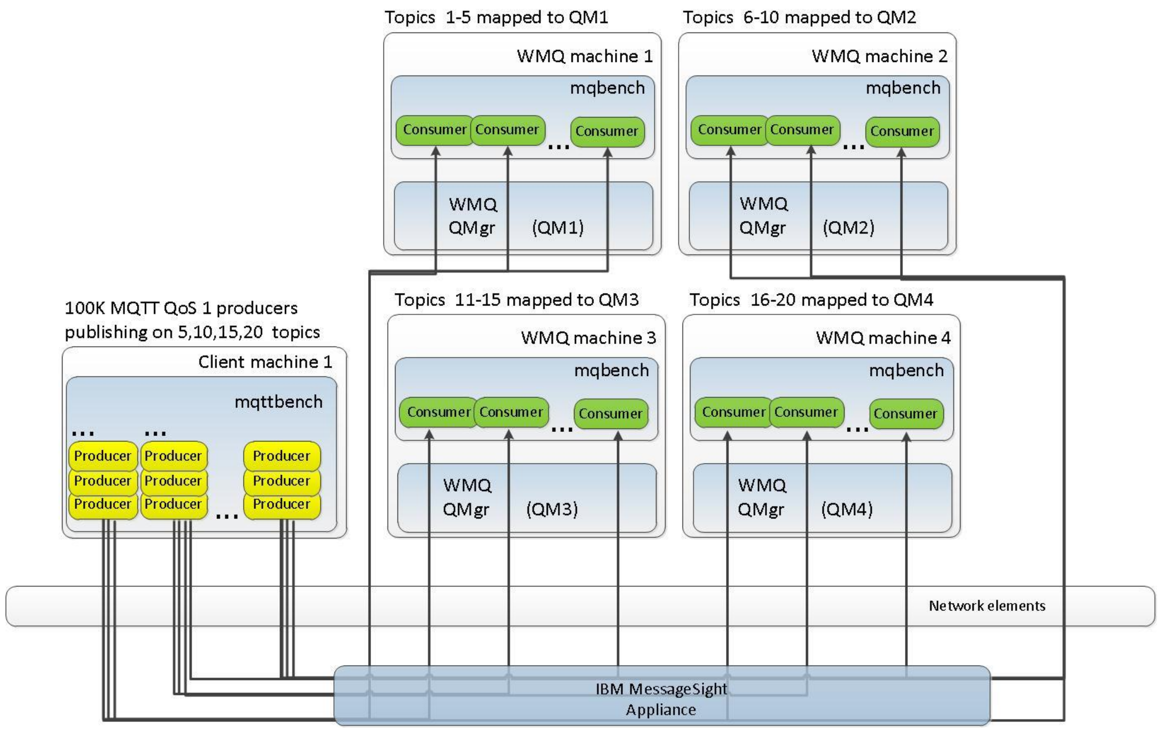
# of MQTT Producers	# of MQTT topics	# of Destination mapping rules	# of MQ topics	# of WMQ QMgrs
100K	5	5	5	1
100K	10	10	10	2
100K	15	15	15	3



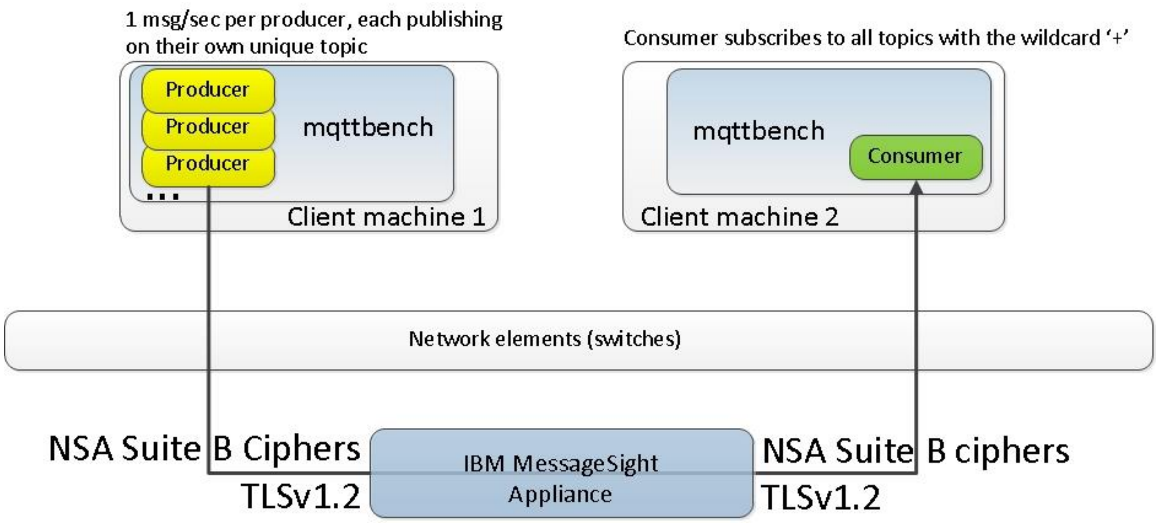
WMQDURABLE

This test case measures the maximum throughput rate at which the IBM MessageSight appliance can forward MQTT QoS 1 messages to a WMQ backend with durable MQ subscribers. The workload, in this test case, is generated from 100K MQTT QoS 1 producers. The workload is evenly distributed across a number of topics, which are mapped, using administratively created destination mapping rules in the appliance, to a number of WMQ queue managers. Thus, the workload is evenly distributed across multiple WMQ queue managers. The destination mapping rules used in this test case are of type, "appliance topic to MQ topic". In this test case a single WMQ queue manager is started on WMQ machine 1. Subsequently, a single instance of *mqbench* is started on WMQ machine 1 and creates durable subscriptions to 5 MQ topics. The entire workload, from all 100K MQTT producers, is forwarded to these 5 MQ topics. On client machine 1, a single instance of *mqttbench* starts 100K MQTT QoS 1 producers and generates the workload, which is distributed across 5 MQTT topics. The destination mapping rules map each of the incoming 5 MQTT topics to a different outgoing MQ topic. The test is then repeated, but with additional WMQ queue managers. The load is distributing evenly across all the WMQ queue managers.

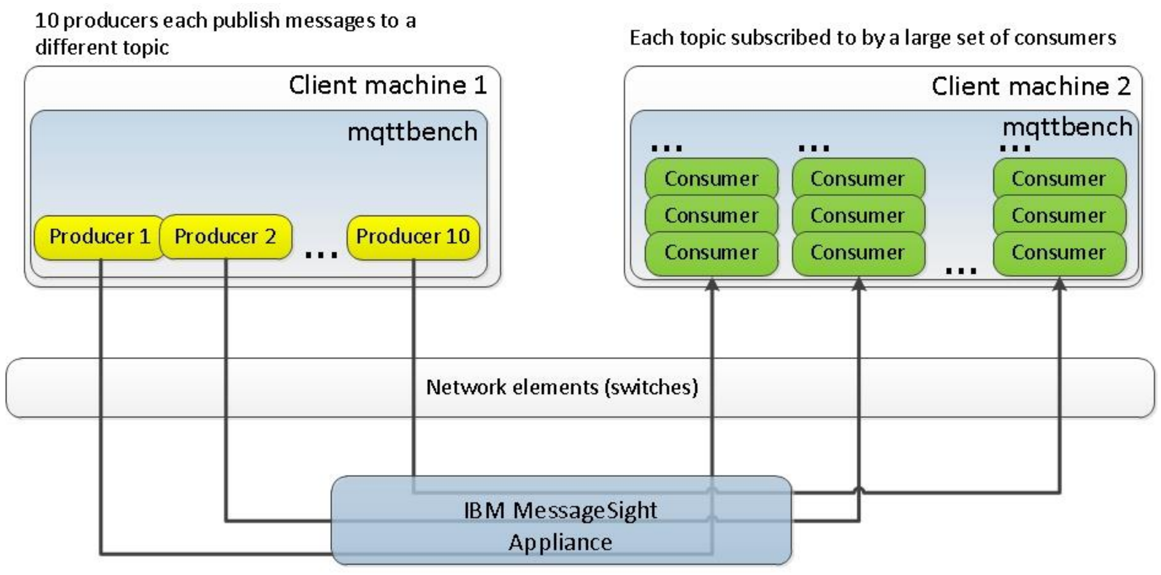
# of MQTT Producers	# of MQTT topics	# of Destination mapping rules	# of MQ durable subscriptions	# of WMQ QMgrs
100K	5	5	5	1
100K	10	10	10	2
100K	15	15	15	3
100K	20	20	20	4



FANIN.SEC
This test case is identical to FANIN, except that communications are secured. In this test case two secure endpoints configured on the IBM MessageSight appliance are used for secure communications. The protocol used in this test case is TLSv1.2 and the cipher category used was BEST (in NSA Suite B ciphers list). The server key used in this test case is 2048 bits. FIPS 140-2 compliance mode is enabled on the appliance. For more detailed information on how the secure endpoints are configured on the IBM MessageSight appliance see Appendix A.


1 msg/sec per producer, each publishing on their own unique topic
Consumer subscribes to all topics with the wildcard '+'
Producer
Producer
Producer
mqttbench
Client machine 1
mqttbench
Consumer
Client machine 2
Network elements (switches)
NSA Suite B Ciphers
TLSv1.2
IBM MessageSight Appliance
NSA Suite B ciphers
TLSv1.2

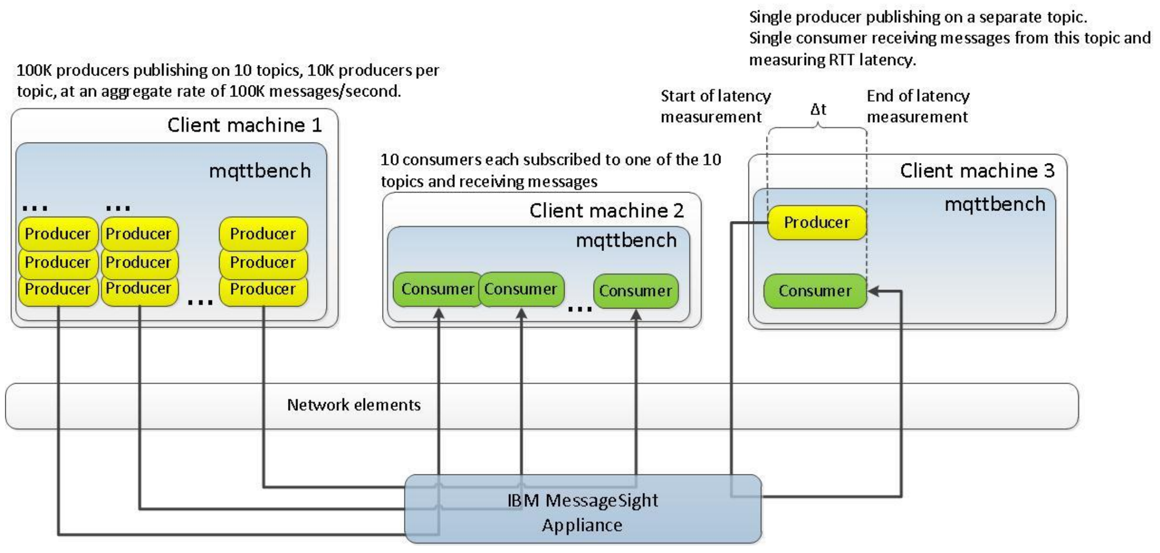
FANOUT.SEC
This test case is identical to FANOUT, except that communications are secured. In this test case two secure endpoints configured on the IBM MessageSight appliance are used for secure communications. The protocol used in this test case is TLSv1.2 and the cipher category used was BEST (in NSA Suite B ciphers list). The server key used in this test case is 2048 bits. FIPS 140-2 compliance mode is enabled on the appliance. For more detailed information on how the secure endpoints are configured on the IBM MessageSight appliance see Appendix A.


10 producers each publish messages to a different topic
Each topic subscribed to by a large set of consumers
Client machine 1
mqttbench
Producer 1
Producer 2
...
Producer 10
Client machine 2
mqttbench
Consumer
Network elements (switches)
IBM MessageSight Appliance

Secure Communications: TLSv1.2 and NSA Suite B Ciphers

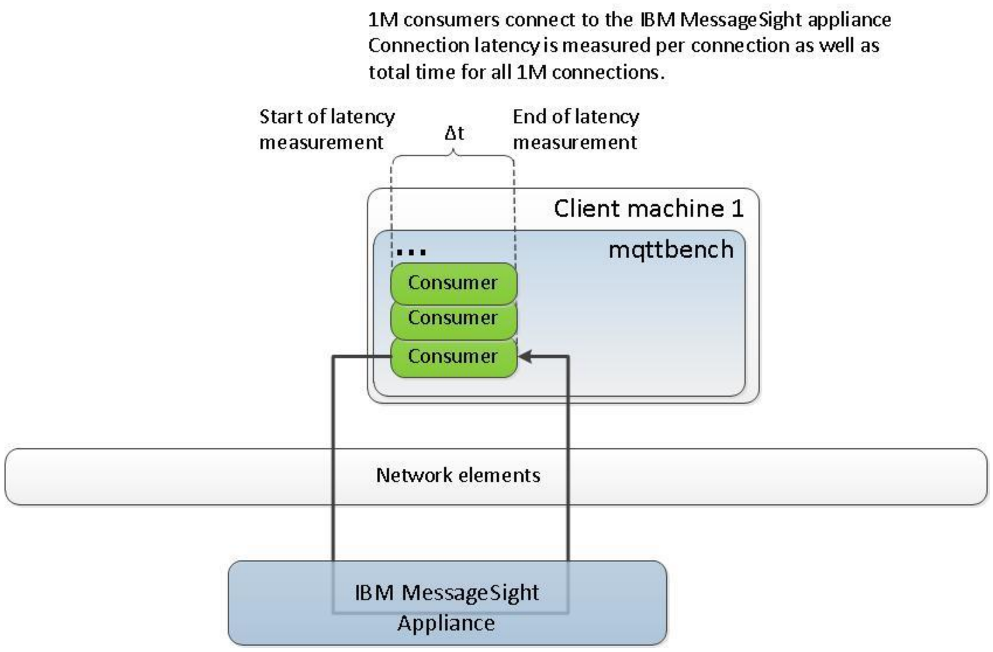
LATENCY

This test case measures application to application latency, while the IBM MessageSight appliance is under load. Application to application latency includes network latency and client OS latency. In this test case 100K MQTT v3.1 producers are publishing messages on 10 topics, that is, 10K producers per topic, at an aggregate rate of 100K message/second. While this workload is running an MQTT consumer and producer pair are started on a separate client machine and are publishing and consuming messages on a separate topic. On client machine 2, a single instance of *mqttbench* starts 10 MQTT (QoS 0) consumers and is ready to consume messages. Subsequently, a second instance of *mqttbench* starts 10K MQTT (QoS 0) producers on client machine 1 and begins generating the producer load, which is consumed by the consumers on machine 2. Finally, a third instance of *mqttbench* is started on client machine 3 which starts both an MQTT (QoS 0) producer and consumer which publish and consume messages on a separate topic. The producer on client machine 3 publishes at a rate of 10K messages/second and the consumer consumes all 10K messages/second. The consumer measures application to application latency on 100% of the messages. Latency measurements are taken over a 5 minute period. The test is repeated for 10K, 50K, and 150K producers sending at an aggregate rate of 10K, 50K, and 150K msgs/sec, respectively, from client machine 1, in order to measure the affect of different loads on latency.


100K producers publishing on 10 topics, 10K producers per topic, at an aggregate rate of 100K messages/second.
Client machine 1
mqttbench
Producer
10 consumers each subscribed to one of the 10 topics and receiving messages
Client machine 2
mqttbench
Consumer
Single producer publishing on a separate topic. Single consumer receiving messages from this topic and measuring RTT latency.
Start of latency measurement
Δt
End of latency measurement
Client machine 3
mqttbench
Producer
Consumer
Network elements
IBM MessageSight Appliance

CONNBURST

This test case measures the total time for the appliance to process a burst of 1M connections from MQTT v3.1 devices. On client machine 1, a single instance of *mqttbench* starts 1M MQTT v3.1 consumers which connect to the appliance. Connection latency includes network latency and client latency.


1M consumers connect to the IBM MessageSight appliance
Connection latency is measured per connection as well as total time for all 1M connections.
Start of latency measurement
Δt
End of latency measurement
Client machine 1
mqttbench
...
Consumer
Consumer
Consumer
Network elements
IBM MessageSight Appliance

Product background

IBM MessageSight appliance:

IBM MessageSight is an appliance-based messaging server that is designed to handle large numbers of connected clients and devices and process high volumes of messages with consistent latency.

Designed to be able to sit at the edge of the internet, IBM MessageSight is the ideal extension of an existing infrastructure, reaching new use cases that include mobile or Internet of Things to connect users, devices, or objects. IBM MessageSight can also be used as a stand-alone server for the next generation of application. The ability to scale to an unprecedented level makes IBM MessageSight ideal to deliver large amounts of data to analytics engines and other big data types of application.

The IBM MessageSight form factor and security features such as authentication, authorization, and SSL/TLS protocols provide a secure point of entry into your enterprise.

IBM MessageSight supports emerging, as well as established, messaging and communication standards, including JMS, WebSockets, and MQTT (an open source lightweight protocol). IBM MessageSight can easily extend an existing WebSphere® MQ infrastructure through MQ Connectivity.

IBM G8316 40GbE switch:

The IBM System Networking RackSwitch G8316 is a 40 Gigabit Ethernet (GbE) aggregation switch designed for the data center, providing speed, intelligence and interoperability on a proven platform.

The RackSwitch G8316 offers up to 16×40 GbE ports, which can also be used as a high-density 10 GbE switch, with 1.28 Tbps—in a 1U footprint. The G8316 provides a cost-efficient way to aggregate multiple racks of servers compared to other expensive core switches, while allowing massive scalability for your data center network. It is an ideal aggregation layer switch when used with the 10/40 GbE RackSwitch G8264 at the access layer.

Designed with top performance in mind, the RackSwitch G8316 provides line-rate, high-bandwidth switching, filtering, and traffic queuing without delaying data. Large data center grade buffers keep traffic moving. Hot-swappable, redundant power and fans along with numerous high availability features enable the RackSwitch G8316 to be available for business-sensitive traffic.

Benchmark participants

The following vendors participated in this benchmark and provided the assets required to complete the benchmark. For the purposes of obtaining the necessary non-disclosure agreements these participants should be contacted:

- IBM

Benchmark assets provided by the participants:

- IBM provided the MessageSight appliance
- IBM configured and optimized the system under test
- IBM provided the 10/40GbE switch
- IBM provided the software and hardware used to generate workloads handled by the IBM MessageSight appliance.
- IBM provided the PDUs used to measure power efficiency during testing
- IBM sponsored and audited the benchmark

Contacts

For questions or to provide feedback regarding these benchmarks or the test results, contact your IBM sales representative.

Methodology

The benchmark specifications used in these tests were developed by IBM and are described above.

Test harness

The software components used to generate the workloads used in these benchmarks were developed by IBM and are listed below.

MQTT workload generator	*mqttbench v1.0*
JMS workload generator	*JMSBenchTest v1.0* *IBM MessageSight JMS client library v1.0*
MQ workload generator	*mqbench v1.0*
Clock source (for latency measurements)	*TSC*

The *mqttbench* test harness program:

- A multi-threaded C program that can simulate up to 1M MQTT v3.1 devices
- Uses TCP, non-blocking communications
- Supports pub/sub messaging. Does not currently support request/response messaging over the same TCP connection.
- Measures latency using the TSC clock source
- Support for SSL/TLS 1.2

The *JMSBenchTest* harness program:

- A multi-threaded Java program which uses the IBM MessageSight JMS client API.
- Supports pub/sub messaging. Does not currently support request/response messaging over the same JMS connection.
- Measures latency using the TSC clock source
- Support for SSL/TLS 1.2

The *mqbench* test harness program:

- A multi-threaded C program which uses the WMQ client API
- Uses WMQ FASTPATH bindings API to communicate with WMQ Queue Managers
- Supports pub/sub messaging

Power measurements

Power measurements taken in these benchmarks were obtained by taking periodic power readings at the outlet level using monitoring facilities of IBM 0U 24 port C13 Switched and Monitored 30A PDUs.

Limitations

- These benchmarks were performed in a controlled laboratory environment in which the mobile and telemetry devices were simulated by the test harness software programs listed above. The test harness programs and the IBM MessageSight appliance were physically connected on the same high speed network switch. As such, the network RTT in this benchmark environment is much smaller than that of real mobile networks.
- The latency metrics used in these benchmarks are round trip measurements which include the latency of the IBM MessageSight appliance, the test harness software programs, and the network switching latency.
- The measurements taken in these benchmarks rely on software instrumentation, which has an impact on performance. This limitation is consistent with all benchmarks.
- The mechanisms used by the test harness programs for controlling the producer message rate can result in an initial message rate spike above the requested message rate before converging on the requested message rate. To compensate for this behavior the test cases are executed by starting the workload at an initial message rate below the maximum achievable message rate and then increasing the message rate until achieving max message rate.

System under test

IBM MessageSight appliance

Firmware version	1.0
Firmware build ID	20130422-2304
Rack units	2U
Power supplies	2 x 750W

Network elements (switches)

Vendor and model	IBM G8316 40GbE RackSwitch (Rear to Front airflow) Layer 2/Layer 3 switch
Software version	IBM Networking OS v7.4.1
Hardware revision	0
Rack units	1U
Power supplies	2 x 450W
Interop guide	ibm.com/networking
Cables	Mellanox FDR 3m DAC QSFP (Part #: MC2207128-003)

PDU

Vendor and model	IBM 0U 24 port C13 switched and monitored 30A single-phase PDUs (Model #: 46M4116)
Receptacle type	24 IEC-320-C13 (10 A)
Plug type	NEMA L6-30P
Software version	IBM DPI V01.06.0005

Client machines

Vendor and model	IBM System X 3650 M3 (Model #: 7945AC1)
Processors	2 x Intel Xeon six-core X5690 3.47GHz
L1/L2/L3 cache	32KB L1 / 256KB L2 / 12MB L3
Memory speed	1333 MHz
Memory size	98GB
PCIe bus speed	PCIe Gen2 x8 lanes
NIC	Mellanox Connect X-3 40GbE NIC
NIC FW/SW	firmware: 2.11.500 software: mlx4_en 1.5.9
BIOS version	IBM uEFI D6E154A (Release date: 09/23/2011)
Rack units	2U
Power supplies	2 x 675W

Client operating system and JVM

OS and version	Red Hat Enterprise Linux Server 6.2 (x86_64)
JVM	IBM J9 JVM 1.7.0 x86_64

WMQ QMgr machines (required for WMQ test cases)

Vendor and model	IBM System X 3650 M3 (Model #: 7945AC1)
Processors	2 x Intel Xeon six-core X5690 3.47GHz
L1/L2/L3 cache	32KB L1 / 256KB L2 / 12MB L3
Memory speed	1333 MHz
Memory size	98GB
PCIe bus speed	PCIe Gen2 x8 lanes
NIC	Mellanox Connect X-3 40GbE NIC
NIC FW/SW	firmware: 2.11.500 software: mlx4_en 1.5.9
BIOS version	IBM uEFI D6E154A (Release date: 09/23/2011)
Rack units	2U
Power supplies	2 x 675W

WMQ QMgr operating system and WMQ software

OS and version	Red Hat Enterprise Linux Server 6.2 (x86_64)
WMQ version	IBM WebSphere MQ v7.5.0.0 (x86_64)

LDAP server machines (required for SECURITY test cases)

Vendor and model	IBM System X 3650 M3 (Model #: 7945AC1)
Processors	2 x Intel Xeon six-core X5690 3.47GHz
L1/L2/L3 cache	32KB L1 / 256KB L2 / 12MB L3
Memory speed	1333 MHz
Memory size	98GB
PCIe bus speed	PCIe Gen2 x8 lanes
NIC	Mellanox Connect X-3 40GbE NIC
NIC FW/SW	firmware: 2.11.500 software: mlx4_en 1.5.9
BIOS version	IBM uEFI D6E154A (Release date: 09/23/2011)
Rack units	2U
Power supplies	2 x 675W

LDAP server operating system and LDAP server software

OS and version	Red Hat Enterprise Linux Server 6.2 (x86_64)
LDAP version	OpenLDAP 2.4.23-20 (x86_64)

Score card

The score card summarizes the results from each benchmark test case performed in this audit. Not all test cases are performed during every audit. The score card lists test case name along side the test results. Detailed descriptions of each test case are provided in the test case description section above. The test results data section below provides additional test result details and charts for those test cases which require additional test results data.

Type	Name	Description	VALUE	AVG	50P	99P	MAX	STDV
Throughput	FANIN	Maximum number of MQTT QoS 0 producers, each sending 1 msg/sec. Message size, 32 bytes. (# of producers)	**400K**					
	FANOUT	Maximum MQTT QoS 0 message rate. Message size, 32 bytes. Fanout ratio 1:100. (Ingress rate msgs/sec) - (Egress rate msgs/sec)	**162K 16.2M**					
	PERSISTENT	Maximum persistent messaging rate (including storing messages to non-volatile media). Message size, 32 bytes. (msgs/sec)	**460K**					
	JMSPUBSUB	Maximum JMS pub/sub messaging rate, quality of service equivalent to MQTT QoS 0. Message size, 32 bytes. (msgs/sec)	**1.4M**					
	JMSQUEUE	Maximum JMS queueing messaging rate using transacted session. Transaction size 100 msgs/txn. Message size, 32 bytes. (msgs/sec)	**800K**					
	JMSPERSISTENT	Maximum persistent messaging rate, using JMS applications. Message size, 100 bytes. (msgs/sec)	**350K**					
	JMSFANOUT	Maximum JMS pub/sub messaging rate, quality of service equivalent to MQTT QoS 0. Message size, 100 bytes. Fanout ratio 1:30 (Ingress rate msgs/sec) - (Egress rate msgs/sec)	**210K 6.3M**					
	WMQPUBSUB	Maximum rate at which MQTT QoS 0 messages can be forwarded to a WMQ backend. Message size, 32 bytes. (msgs/sec)	**230K**					
	WMQDURABLE	Forwarding rate of MQTT QoS 1 messages to a total of 4 WMQ queue managers with durable MQ subscribers. Transaction size of durable MQ clients is 200 msgs/transaction. Message size, 32 bytes. (msgs/sec)	**92K**					
	FANIN.SEC	FANIN test, but with secure communications: TLSv1.2, cipher category BEST (NSA suite B), and 2048 bit key. Message size, 32 bytes. (# of producers)	**400K**					
	FANOUT.SEC	FANOUT test, but with secure communications: TLSv1.2, cipher category BEST (NSA suite B), and 2048 bit key. Message size, 32 bytes. Fanout ratio 1:10. (Ingress rate msgs/sec) - (Egress rate msgs/sec)	**330K 3.3M**					
Latency	LATENCY	Application to application latency while the appliance is under load, that is, handling 150K producers each publishing at 1 msg/sec. Message size, 32 bytes. Message rate, 10K msgs/sec. (microseconds)		**84**	**83**	**144**	**427**	**25**
Connection rate	CONNBURST	Total time to process 1M connections from MQTT device. (seconds)	**65**					

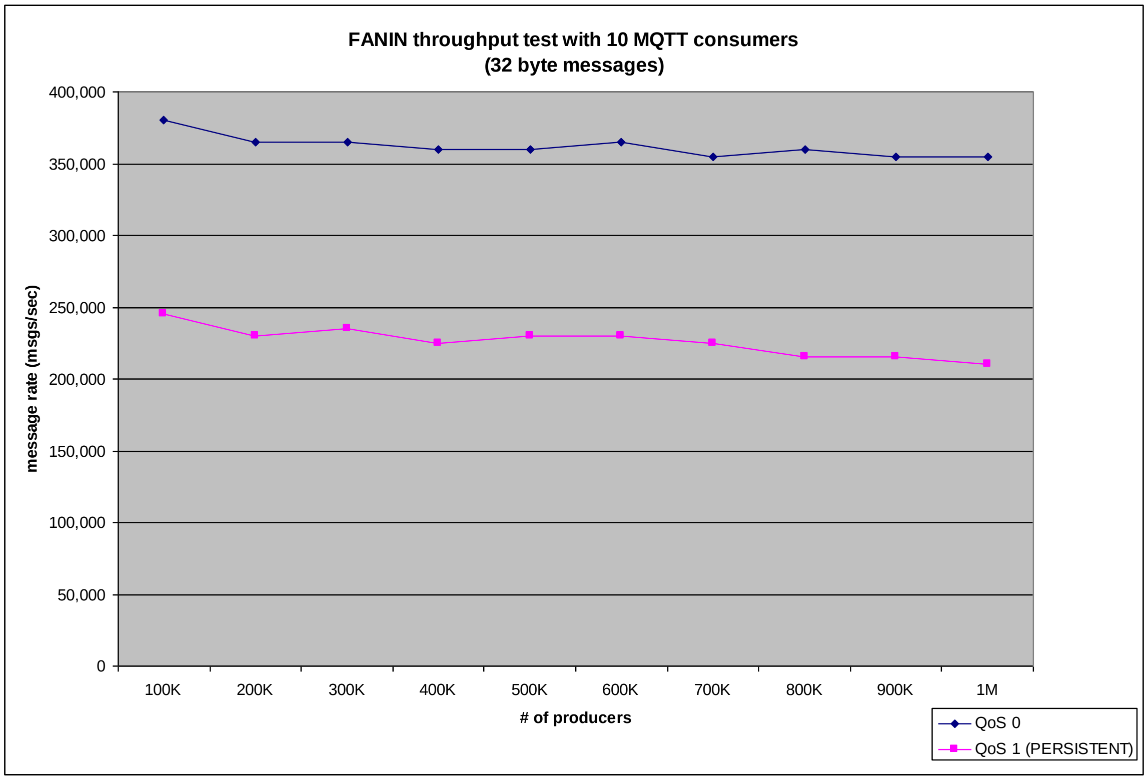
Data-center Efficiency	PERSISTENT.MPW	Power consumption by the appliance while storing messages to non-volatile media at peak rate. PERSISTENT test result, divided by average Watts/sec. The overall power consumption of the appliance calculated as the sum of the average consumption for power supply #1 and power supply #2. (msgs/Watt)	**1523**					
	PERSISTENT.MPU	PERSISTENT test result, divided by rack units (msgs/sec/U)	**230K**					

Additional test results data

Additional test results data and charts supporting the claims made in the score card are provided here.

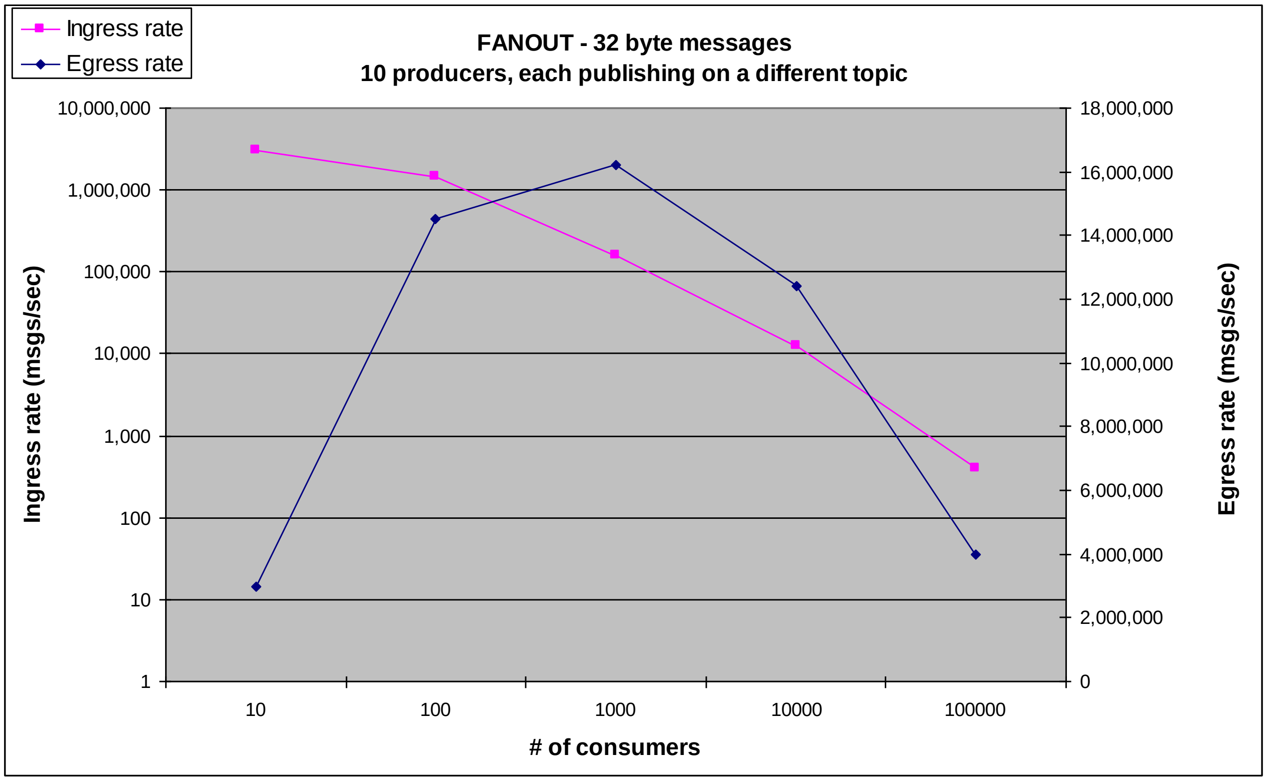
FANIN
The high water mark (HWM) of the subscription in this test was 277 messages.

The additional FANIN test result data below shows that the maximal message rate handled by the appliance remains fairly constant whether receiving messages from 100K MQTT producer clients or 1M.



FANOUT
The maximum MQTT QoS 0 message rate was achieved with a fanout ratio of 1:100. That is, when 100 consumers were subscribed to each of the 10 topics for a total of 1000 consumers. The chart below shows how ingress (into the appliance) and egress (out of the appliance) throughput changes as more consumers are subscribed and receiving messages.

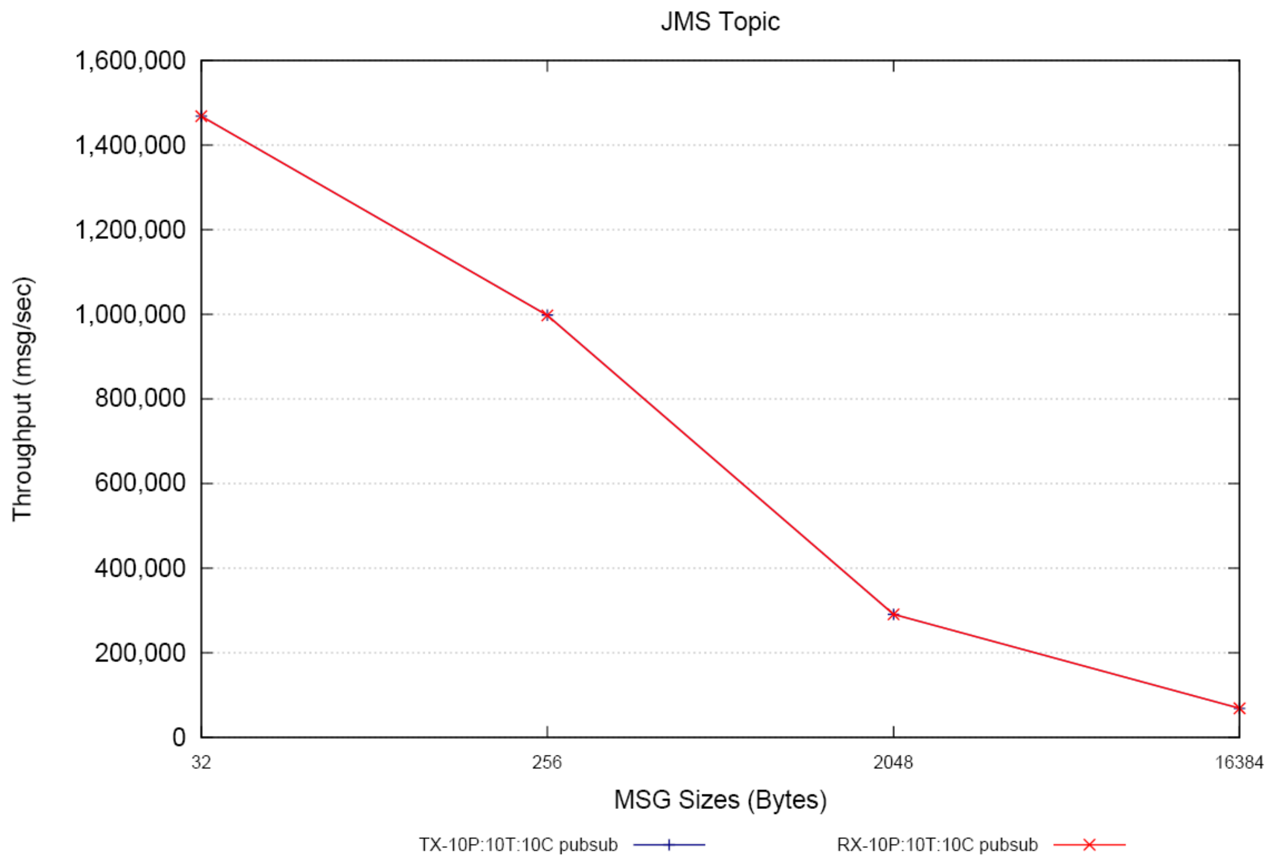
NOTE: the ingress (or incoming) message rate is plotted against the left hand Y-axis and the egress (or outgoing) message rate is plotted against the right hand Y-axis. Also note that the left hand Y-axis is a logarithmic scale.


Ingress rate
Egress rate
FANOUT - 32 byte messages
10 producers, each publishing on a different topic
Ingress rate (msgs/sec)
Egress rate (msgs/sec)
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
18,000,000
16,000,000
14,000,000
12,000,000
10,000,000
8,000,000
6,000,000
4,000,000
2,000,000
0
10
100
1000
10000
100000
# of consumers

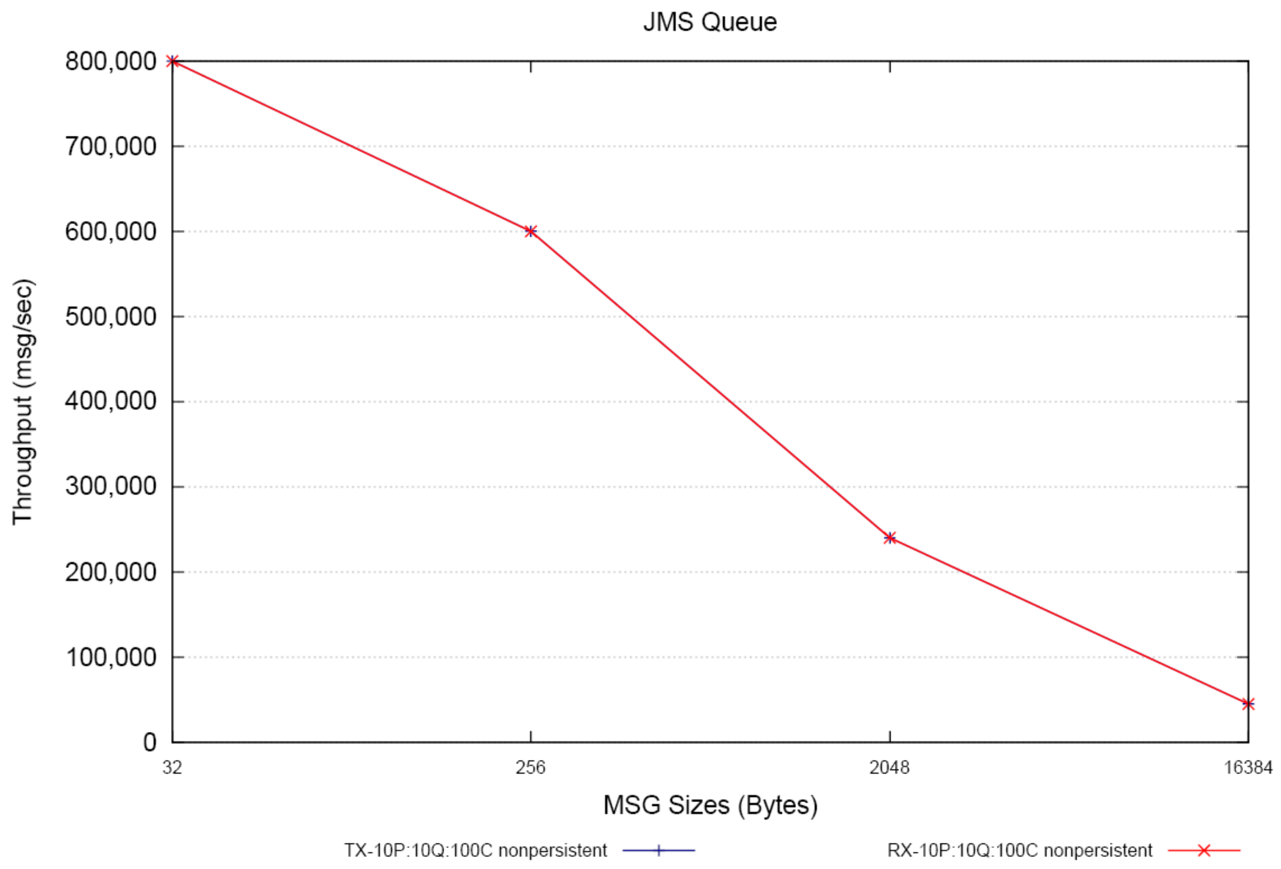
PERSISTENT
The high water mark (HWM) of all 10 subscriptions in this test was 137 messages.

JMSPUBSUB
In addition to the 32 byte message test result the chart below also shows JMS pub/sub message rates for larger messages.

NOTE: TX is the aggregate producer message rate. RX is the aggregate consumer message rate.

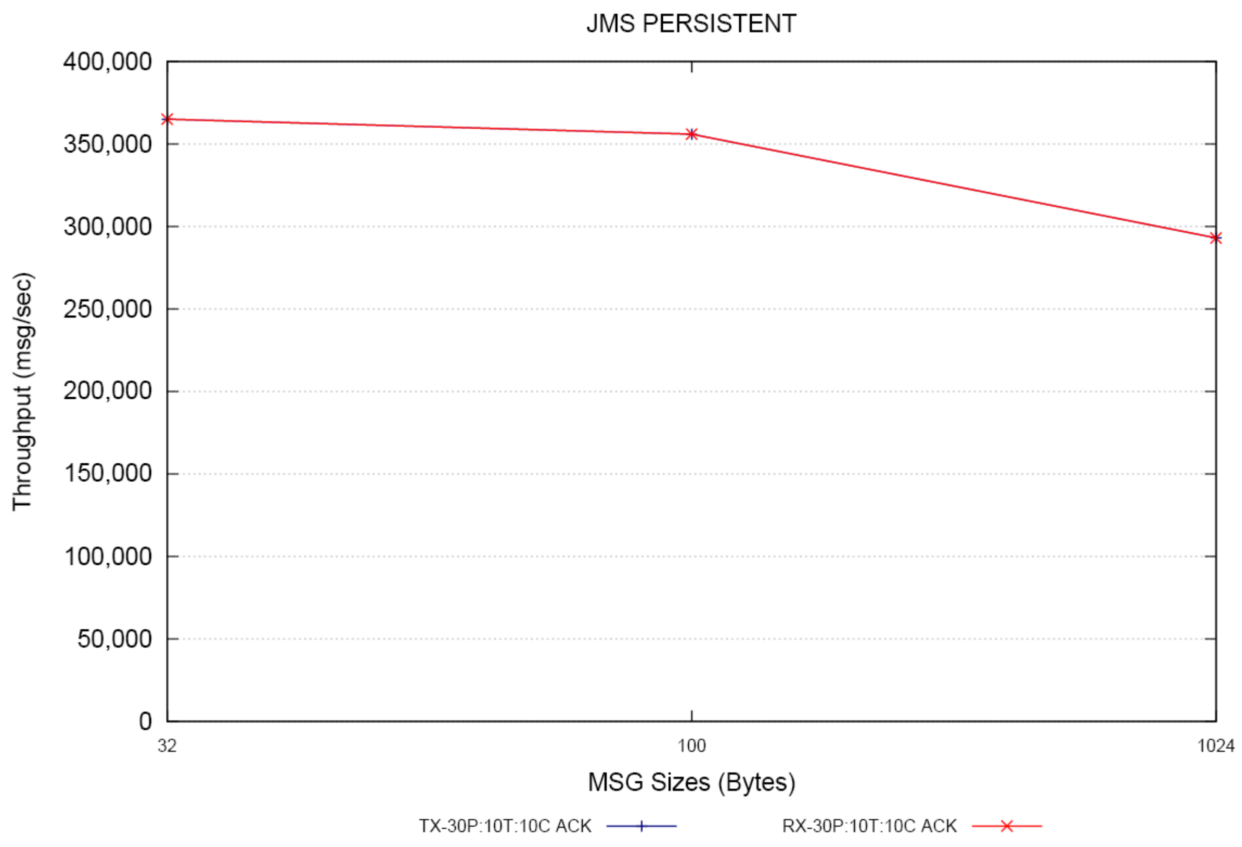


JMSQUEUE
In addition to the 32 byte message test the chart below also shows JMS queueing message rates for larger messages.



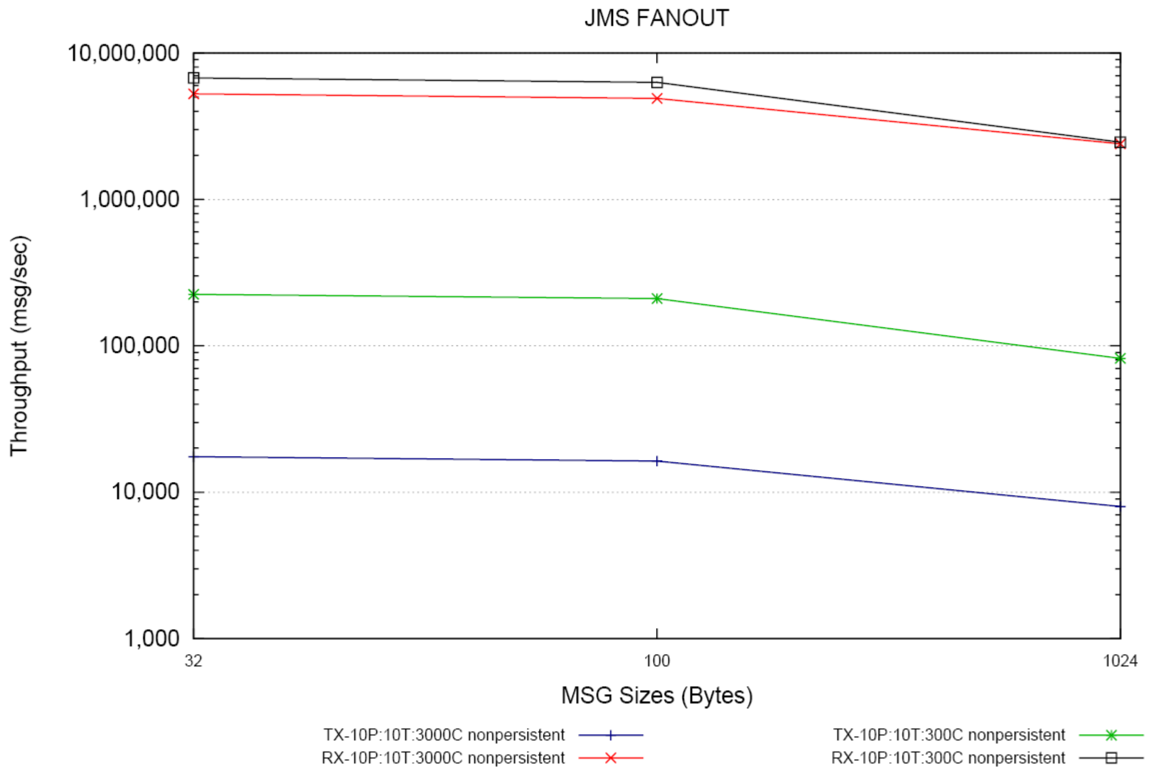
JMSPERSISTENT

In addition to the 100 byte message test the chart below also shows JMS persistent message rates for other message sizes.



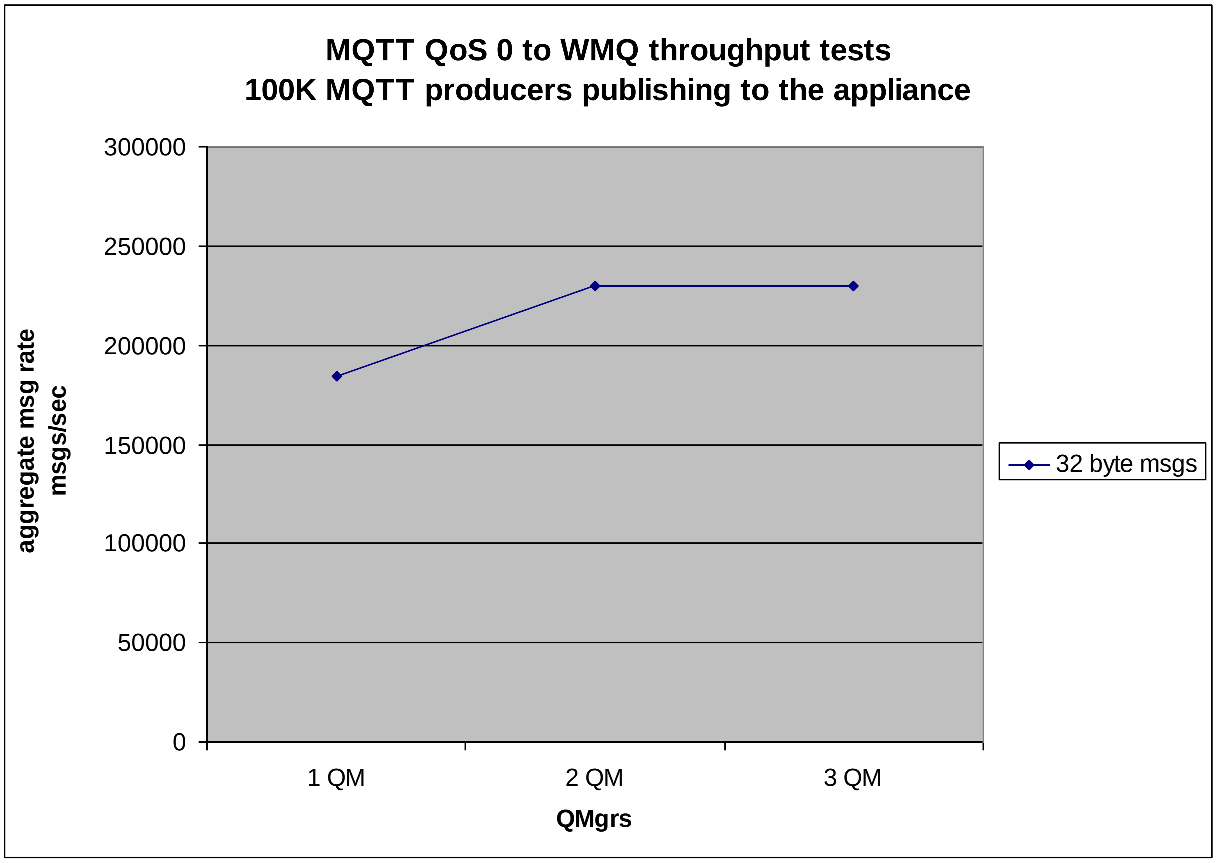
JMSFANOUT

In addition to the 100 byte message test the chart below also shows JMS fanout message rates for other message sizes and fanout ratios.

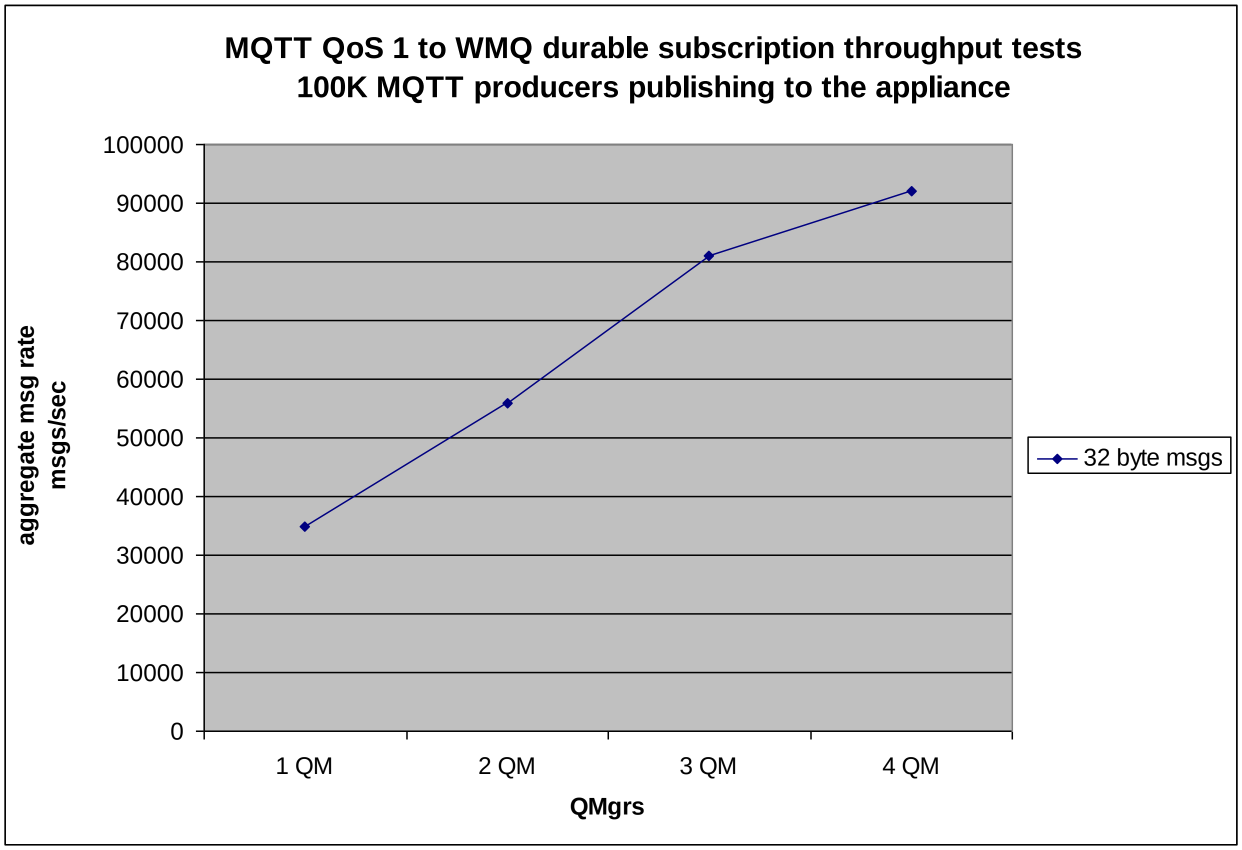


WMQPUBSUB

The chart below shows the forwarding rate of the IBM MessageSight appliance as additional WMQ queue managers are added to handle the workload. The high water mark (HWM) of all the subscriptions in this test was less than 10 messages. In the single queue manager test the queue manager is the bottleneck.



WMQDURABLE
The chart below shows the forwarding rate of the IBM MessageSight appliance as additional WMQ queue managers are added to handle the workload. The high water mark (HWM) of all the subscriptions in this test was less than 6K messages. In all tests the disk subsystem of the WMQ queue manager machines was the bottleneck, that is, messages in the appliance began accumulating when disk utilization on the WMQ queue manager machines reached 100%.

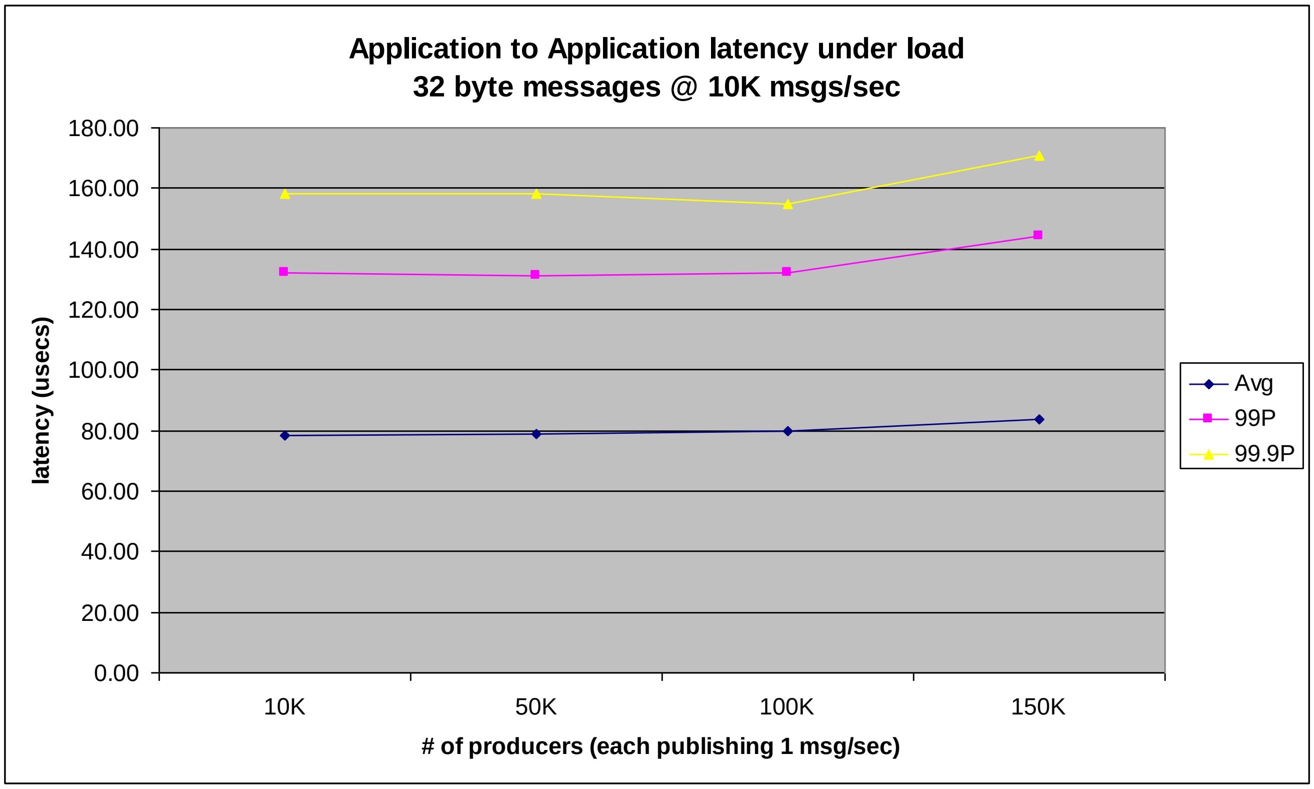


FANIN.SEC
The high water mark (HWM) of the subscription in this test was 170 messages.

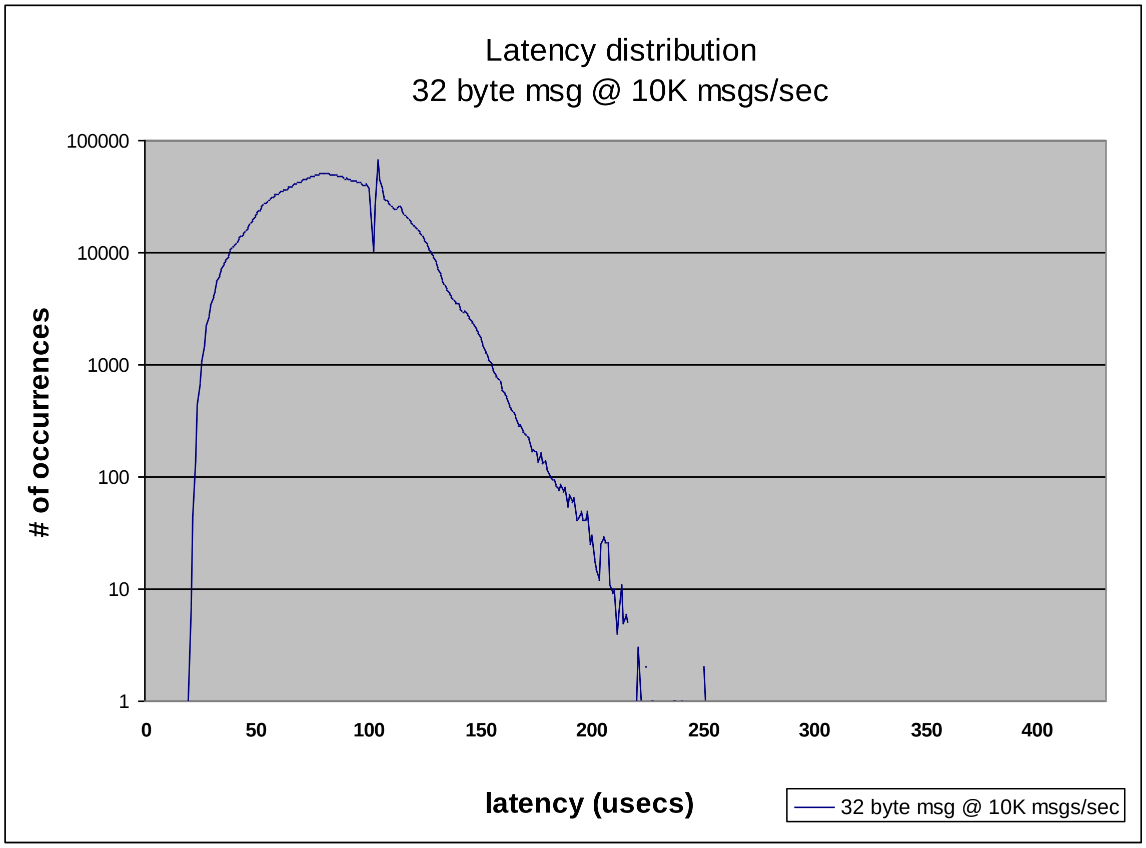
FANOUT.SEC
The maximum MQTT QoS 0 message rate was achieved with a fanout ratio of 1:10. That is, when 10 consumers were subscribed to each of the 10 topics for a total of 100 consumers.

Fanout ratio	# of topics	# of consumers	# of producers	Ingress rate (msgs/sec)	Egress rate (msgs/sec)
1:1	10	10	10	2.5M	2.5M
1:10	10	100	10	330K	3.3M
1:100	10	1000	10	20K	2M

LATENCY
The chart below shows how average latency is affected by changes in the load that the appliance is under.



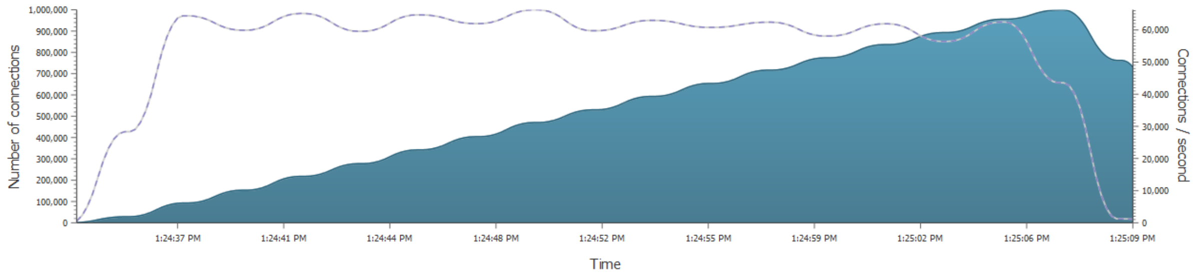
The chart below shows the latency distribution with a background workload of 150K producers publishing at 1 message per second.



CONNBURST

The chart below is from the MessageSight WebUI for connection monitoring. It shows the appliance processing 1M MQTT connections.

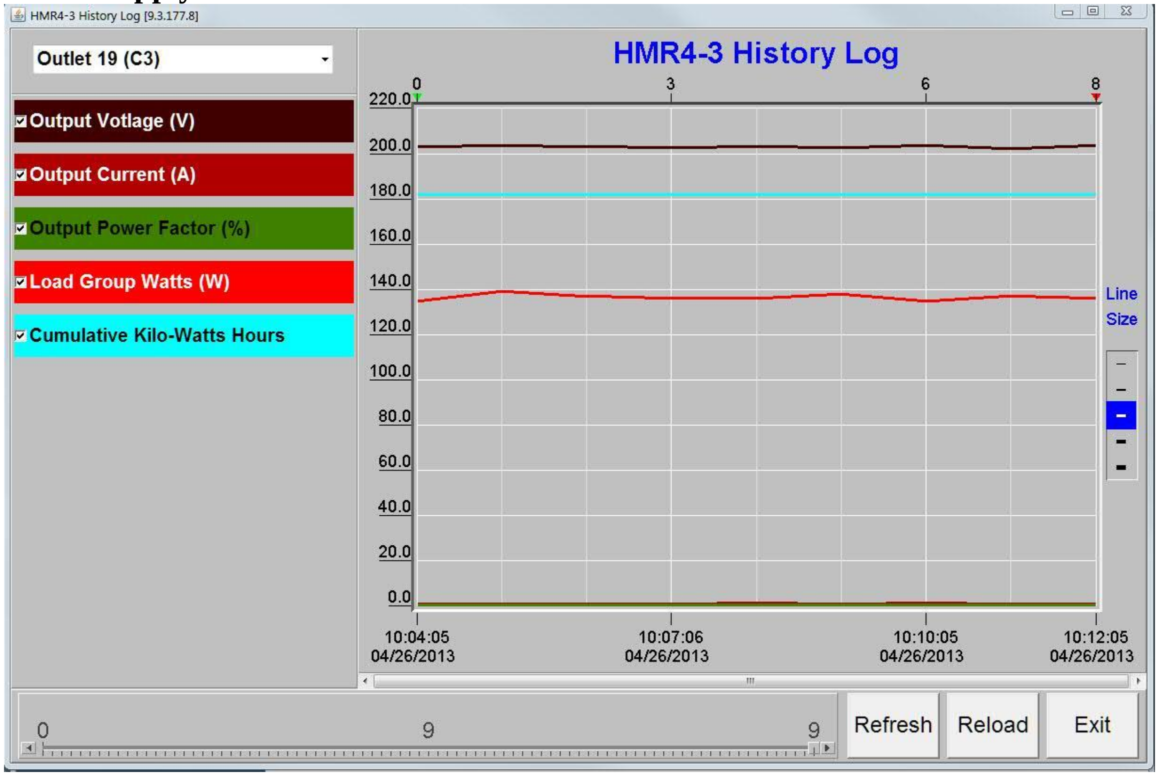
Note: that the chart below does not capture the entire test window.



PERSISTENT.MPW

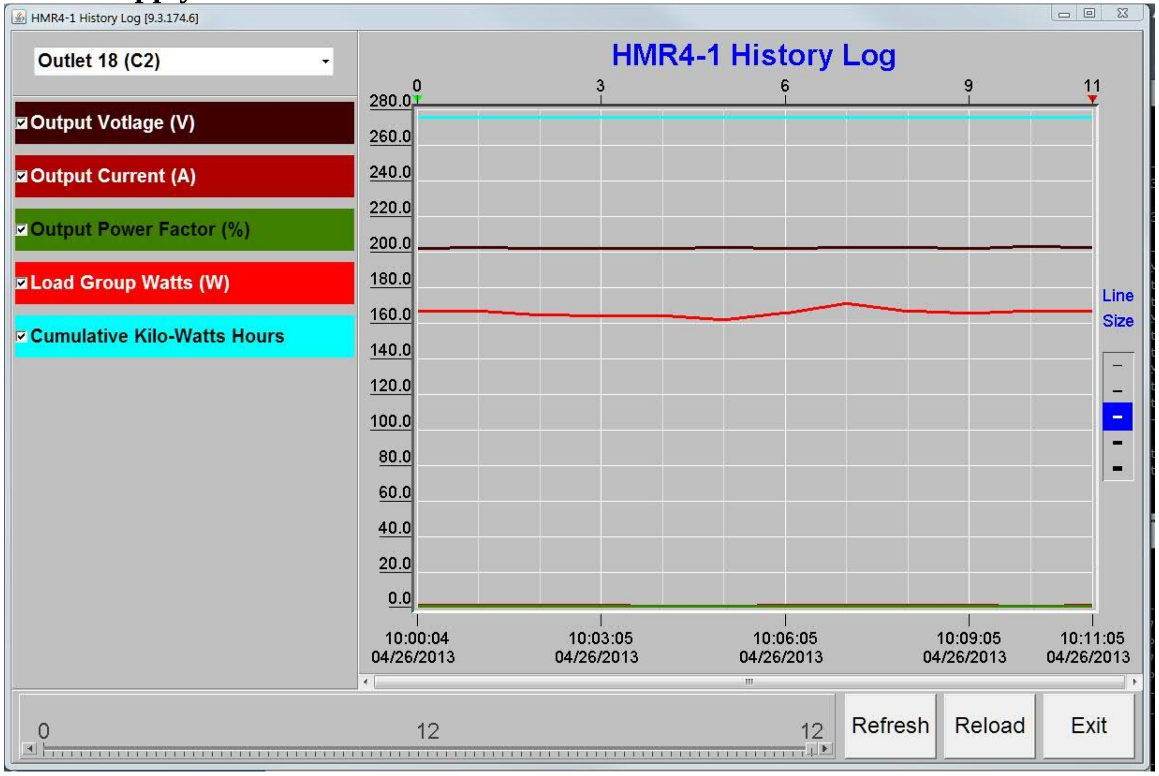
The following graphs were captured during the PERSISTENT test case at peak throughput. The data was captured using the monitoring features of the IBM 0U 24 port C13 Switched and Monitored 30A PDUs. Separate graphs are shown below for both redundant power supplies on the IBM MessageSight appliance. The power measurements were taken over an 8 minute period and show power consumption at the individual outlet receptacle level.

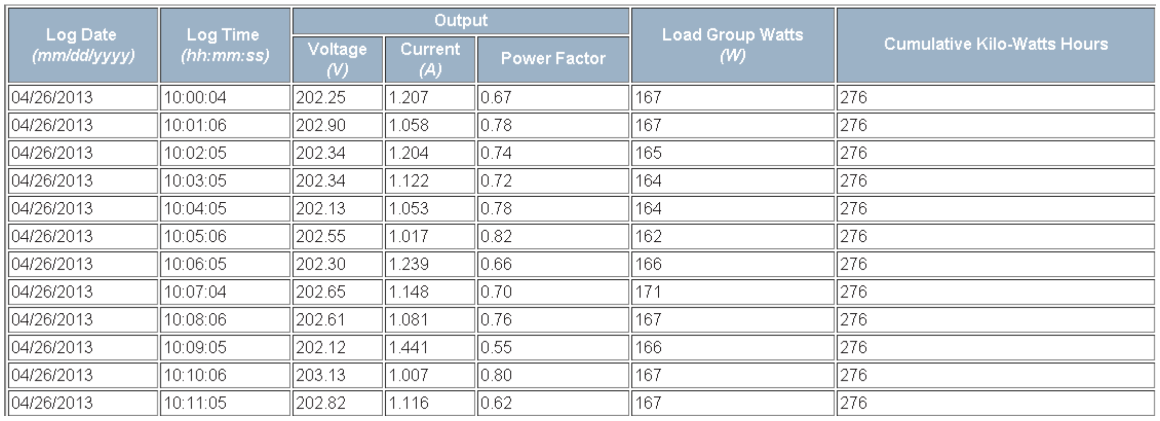
Power Supply #1



Log Date (mm/dd/yyyy)	Log Time (hh:mm:ss)	Output			Load Group Watts (W)	Cumulative Kilo-Watts Hours
		Voltage (V)	Current (A)	Power Factor		
04/26/2013	10:04:05	203.18	1.169	0.58	135	182
04/26/2013	10:05:05	203.55	0.987	0.69	139	182
04/26/2013	10:06:05	203.14	1.150	0.60	137	182
04/26/2013	10:07:06	202.85	0.890	0.49	136	182
04/26/2013	10:08:05	203.05	1.501	0.50	136	182
04/26/2013	10:09:06	202.63	1.033	0.67	138	182
04/26/2013	10:10:05	203.46	1.444	0.49	135	182
04/26/2013	10:11:06	202.28	1.202	0.56	137	182
04/26/2013	10:12:05	203.71	0.884	0.77	136	182

Power Supply #2





Log Date (mm/dd/yyyy)	Log Time (hh:mm:ss)	Output			Load Group Watts (W)	Cumulative Kilo-Watts Hours
		Voltage (V)	Current (A)	Power Factor		
04/26/2013	10:00:04	202.25	1.207	0.67	167	276
04/26/2013	10:01:06	202.90	1.058	0.78	167	276
04/26/2013	10:02:05	202.34	1.204	0.74	165	276
04/26/2013	10:03:05	202.34	1.122	0.72	164	276
04/26/2013	10:04:05	202.13	1.053	0.78	164	276
04/26/2013	10:05:06	202.55	1.017	0.82	162	276
04/26/2013	10:06:05	202.30	1.239	0.66	166	276
04/26/2013	10:07:04	202.65	1.148	0.70	171	276
04/26/2013	10:08:06	202.61	1.081	0.76	167	276
04/26/2013	10:09:05	202.12	1.441	0.55	166	276
04/26/2013	10:10:06	203.13	1.007	0.80	167	276
04/26/2013	10:11:05	202.82	1.116	0.62	167	276

Appendix A: IBM MessageSight appliance configuration

Unless called out explicitly in the sections below all appliance settings were left as the default appliance settings.

Network interface configuration

Both 40GbE network interfaces were used for in these benchmarks. Each network interface was configured with 9 IPv4 addresses for a total of 18 IPv4 addresses. Each IPv4 address was configured with a 23 bit subnet mask and each IPv4 address is in a different IP subnet.

eth6 - subnets (10.10.0.0/23, 10.10.2.0/23, 10.10.4.0/23, etc.)
eth7 - subnets (10.12.0.0/23, 10.12.2.0/23, 10.12.4.0/23, etc.)

The network interfaces on the client machines were configured in the same fashion.

Messaging Hub configuration

One connection policy with the default connection policy settings was created. One messaging policy for topics and another messaging policy for queues were created. Both messaging policies specified the wildcard destination '*'. Both messaging policies set a limit of 1M messages to accumulate per subscription. Four endpoints were created, two secure (16903 and 16904) and two non-secure (16901 and 16902). SSL and SECURITY test cases used the secure endpoints. For all other test cases the two non-secure endpoints were used.

Security profile configuration

A single security profile was created. TLSv1.2 is specified as the minimal protocol method and the cipher category used is BEST (in NSA Suite B ciphers list). FIPS 140-2 compliance is enabled.

Certificate profile configuration

A single certificate profile was created. The server certificate and key used in all SSL and SECURITY test cases is 2048 bits.

MQ Connectivity configuration

For the WMQPUBSUB test case, a one to one mapping of incoming MQTT topics to outgoing MQ topics is used. The rule type used is "Appliance topic to MQ topic". The max messages field is set to 100K. Source is t.000000, where X is 1-15. Destination is ima2mqX, where X is 1-15.

Appendix B: Client machine tuning

Kernel tuning parameters

The following kernel parameters were set on the client machines for these benchmarks.

Name	Value
fs.nr_open	2000000
net.core.wmem_max	65536
net.core.rmem_max	8388608
net.core.wmem_default	16384
net.core.rmem_default	16384
net.core.netdev_max_backlog	2097152
net.ipv4.ip_local_port_range	5000 65000
net.ipv4.tcp_wmem	2048 16384 65536
net.ipv4.tcp_rmem	4096 16384 8388608
net.ipv4.tcp_mem	65536 8388608 16777216
net.ipv4.tcp_tw_reuse	1
net.ipv4.tcp_timestamps	1
net.ipv4.tcp_window_scaling	1
net.ipv4.tcp_sack	1
net.ipv4.tcp_synack_retries	10
net.ipv4.tcp_keepalive_intvl	15
net.ipv4.tcp_keepalive_probes	5
net.ipv4.tcp_fin_timeout	15
kernel.threads-max	2000000
kernel.pid_max	2000000
vm.nr_hugepages	500
vm.max_map_count	4000000

NIC tuning parameters

The following NIC tuning parameters were set on the client machines for these benchmarks.

Name		Value
RX ring buffers		8192
TX ring buffers		8192
txqueuelen		40000
MTU		1500
Device Active MTU		1024
Interrupt coalescence	adaptive-rx	off
	rx-usecs	10
	rx-frames	0
	tx-usecs	0
	tx-frames	0

User and process limits

The following user and process limits were set on the client machines for these benchmarks.

Name		Value
nofile		2000000
stack	soft	4096
	hard	32768
nproc		unlimited
core		unlimited
memlock		unlimited
rtprio		100

OS services stopped

The following operating system services were stopped on the client machines before executing these benchmarks. The client operating system was configured to run at runlevel 3. All firewall related services were disabled for these benchmarks, due to the large impact on packet throughput in the Linux TCP stack.

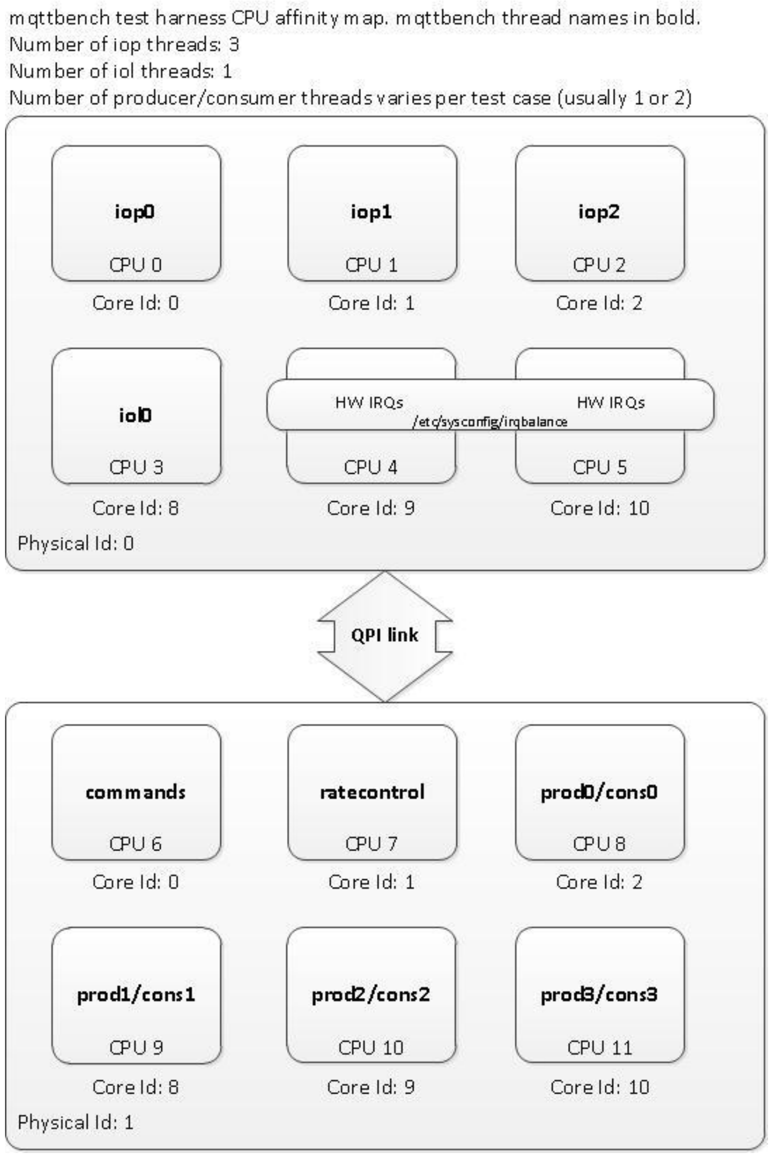
Services stopped: iptables, ip6tables, cpuspeed, cups, crond, atd, auditd, autofs, rhnsd, rhsmcertd, postfix, mdmonitor, jexec, netfs, and more

Interrupt and application CPU affinity tuning parameters

The irqbalance operating system service was configured to pin all non-local interrupts (including network interrupts) to CPUs 4 and 5.

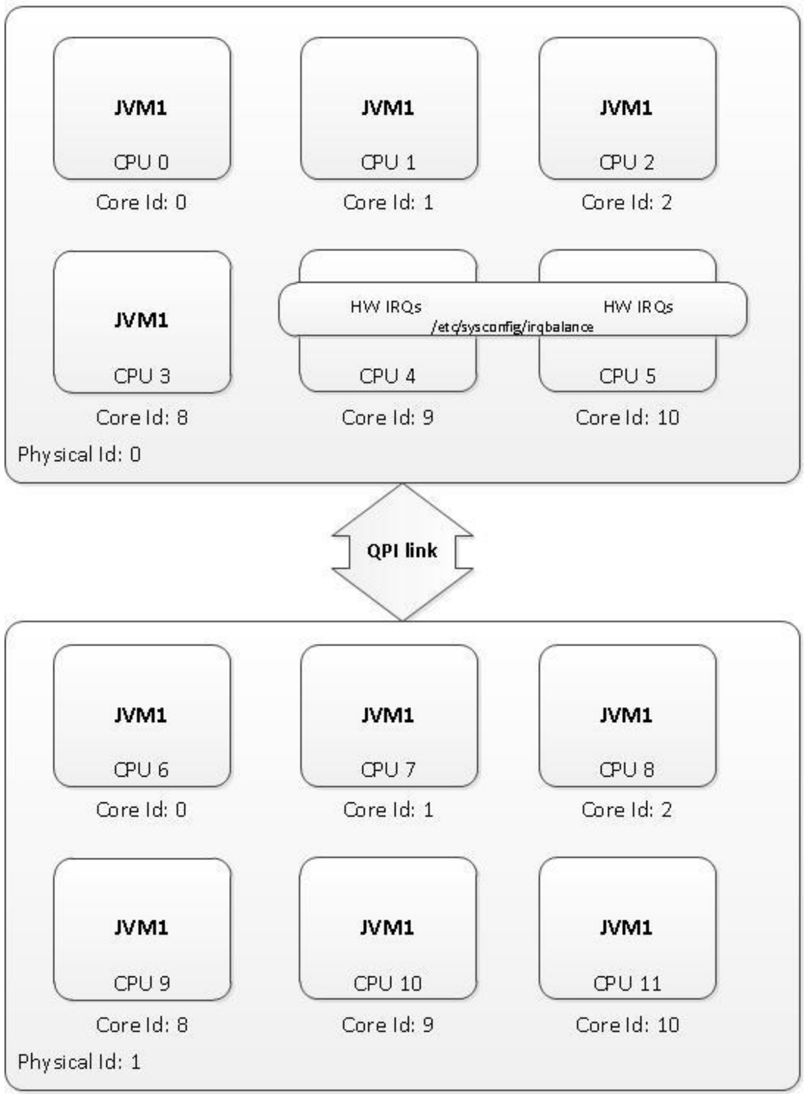
mqttbench affinity

The following diagram shows how the *mqttbench* test harness threads were pinned to the CPUs of the client machine.



JMSBenchTest affinity
The following diagram shows how the *JMSBenchTest* test harness threads were pinned to the CPUs of the client machine.

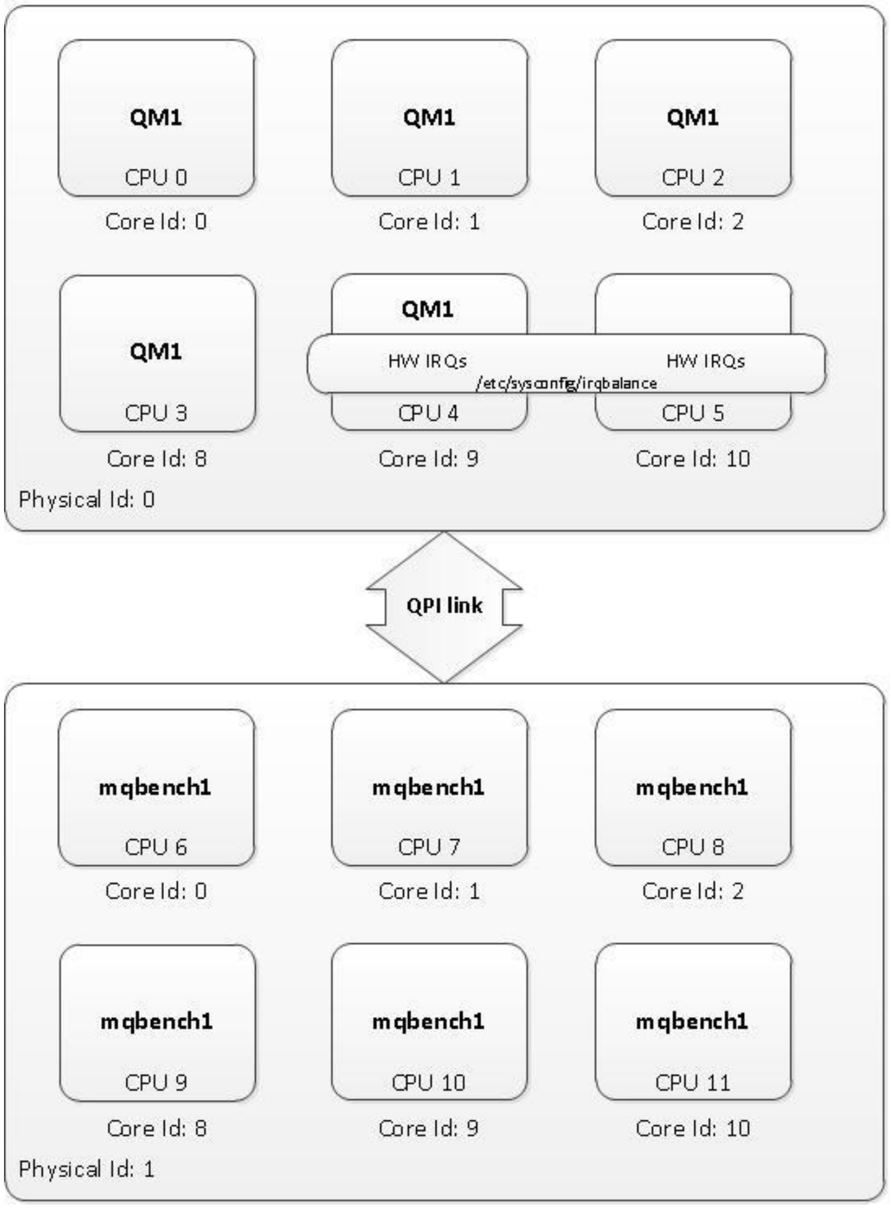
JMSBenchTest test harness CPU affinity map. Single JVM pinned to all CPUs except CPUs 4 and 5.



WMQ QMgr and mqbench affinity

The following diagram shows how the WMQ queue manager and the *mqbench* test harness threads were pinned to the CPUs of the client machine.

WMQ QMgr and mqbench test harness CPU affinity map. A single WMQ Qmgr was pinned to CPUs 0-4. A single instance of mqbench was pinned to CPUs 6-11.



JVM tuning parameters

The following JVM tuning parameters were set for the JMS benchmark tests.

Name	**Value**
Initial Java heap size	-Xms20G
Max Java heap size	-Xmx20G
Max stack size for Java threads	-Xss256K
GC policy	-Xgcpolicy:metronome
Number of GC threads	-Xgcthreads5
GC tuning	-XX:+UseParallelGC
JIT tuning	-Xjit:optLevel=hot,count=1000

Appendix C: WMQ QMgr tuning

The details of how the WMQ queue managers were tuned for the WMQ test cases are shown below.

Kernel tuning parameters

The following kernel parameters were set on the QMgr machines for these benchmarks

Name	Value
kernel.shmall	5368709120
kernel.shmmax	5368709120
kernel.sem	512 524288 256 4096

Queue manager tuning parameters

The following queue manager parameters were set on QMgr machines for these benchmarks.

Name	Value
Max handles per connection (-h)	50000
MQIBindType	FASTPATH
MaxChannels	5000
MaxActiveChannels	5000
PubSubCacheSize	48
SubscriberHashTableRows	10243
TopicHashTableRows	10243
LogBufferPages	512
SubNameHashTableRows	10243
DefaultQBufferSize	1048576
DefaultPQBufferSize	1048576

Queue/Topic tuning parameters

The following queue/topic parameters were set for these benchmarks.

Name	Value
SVRCONN Channel max messages	104857600
QMgr max messages	104857600
System model queue max messages	104857600
System base topic response type	asynchronous
SVRCONN Channel (sharecnv)	1

Appendix D: Test harness execution details

The details of how the test harness programs were executed for each test case, including command line parameters and environment details are shown below.

FANIN

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -tx 0:0:/t/t:1:200000:2:0 -r 100000 -s 32-32 -wr

From mqttbench command line increase aggregate rate to 400K msgs/sec
```

Machine 2

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=0
export IMAPort="16901"
export IMAServer="10.10.0.222"

./imaclnt.sh -d 0 -rx 0:1:/t/+:1:1:1:0
```

FANOUT

Machine 1

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

./imaclnt.sh -d 0 -tx 0:0:t:10:5:2:0 -r 100000 -s 32-32 -wr -rrs
```

From *mqttbench* command line increase aggregate rate until appliance can no longer handle the incoming message rate.

Machine 2

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -rx 0:1:t:10:500:2:0 -rrs
```

PERSISTENT

Machine 1

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

./imaclnt.sh -d 0 -tx 1:0:t:10:5:2:0 -rrs -wr -r 100000 -s 32-32
```

From *mqttbench* command line increase aggregate rate to 460K msgs/sec

Machine 2

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

./imaclnt.sh -d 0 -rx 1:0:t:10:5:2:0 -rrs
```

JMSPUBSUB

Machine 1

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export AsyncSend=1 (IBM MessageSight JMS client connection factory
parameter: AllowAsynchronousSend)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -tx -1:0:0:t:1:1:1:10:0 -r
1400000 -ps -s 32-32 -t 1
```

Machine 2

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export ClientMessageCache=100000 (IBM MessageSight JMS client
connection factory parameter: ClientMessageCache)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -rx -1:0:3:2:0:t:1:1:1:10:0 -ps -
s 32-32
```

JMSQUEUE

Machine 1

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export AsyncSend=1 (IBM MessageSight JMS client connection factory
parameter: AllowAsynchronousSend)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -tx -1:100:0:q:1:1:1:10:0 -r
800000 -s 32-32 -t 1
```

Machine 2

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export ClientMessageCache=100000 (IBM MessageSight JMS client
connection factory parameter: ClientMessageCache)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -rx -1:100:3:2:0:q:1:1:10:-10:0 -
s 32-32
```

JMSPERSISTENT

Machine 1, 2, 3

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK= (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -tx -1:0:1:t:1:1:1:10:0 -ps -r
120000 -s 100-100
```

Machine 4

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK= (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export ClientMessageCache=100000 (IBM MessageSight JMS client
connection factory parameter: ClientMessageCache)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -rx -1:0:3:2:1:t:1:1:1:10:0 -ps -
s 100-100
```

JMSFANOUT

Machine 1

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export AsyncSend=1 (IBM MessageSight JMS client connection factory
parameter: AllowAsynchronousSend)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -tx -1:0:0:t:1:1:1:10:0 -r 17500
-ps -s 32-32 -t 1
```

Machine 2, 3, 4

```
export IMAClient=JMS
export LargeConn=0
export BatchingDelay=1
export DisableACK=1 (IBM MessageSight JMS client connection factory
parameter: DisableACK)
export ClientMessageCache=100000 (IBM MessageSight JMS client
connection factory parameter: ClientMessageCache)
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

taskset -c 0-3,6-11 ./imaclnt.sh -d 0 -rx -1:0:3:2:0:t:1:1:-100:10:0 -
ps -s 32-32
```

WMQPUBSUB

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"
```

Single QMgr test

```
./imaclnt.sh -d 0 -tx 0:0:t:5:50000:2:0 -r 100000 -s 32-32 -wr -rrs
```

Two QMgr test

```
./imaclnt.sh -d 0 -tx 0:0:t:10:50000:2:0 -r 100000 -s 32-32 -wr -rrs
```

Three QMgr test

```
./imaclnt.sh -d 0 -tx 0:0:t:15:50000:2:0 -r 100000 -s 32-32 -wr -rrs
```

From *mqttbench* command line increase aggregate rate to target rate.

Machine 2

Start the WMQ QMgr

```
./crtqms.sh 1 1 50000 0-4
```

Start mqbench

```
export IMAClient=MQ
export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf
true -jb QM1 -jt mqb -d ima2mq -db 1 -dx 5 -rl 0 -nt 5
```

Machine 3

Start the WMQ QMgr

```
./crtqms.sh 1 1 50000 0-4
```

Start mqbench

```
export IMAClient=MQ
export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf
true -jb QM1 -jt mqb -d ima2mq -db 6 -dx 10 -rl 0 -nt 5
```

Machine 4

Start the WMQ QMgr

```
./crtqms.sh 1 1 50000 0-4
```

Start mqbench

```
export IMAClient=MQ
export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf
true -jb QM1 -jt mqb -d ima2mq -db 11 -dx 15 -rl 0 -nt 5
```

WMQDURABLE

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901 16902
16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222 10.10.4.222
10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222 10.10.10.222
10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222 10.12.14.222 10.12.16.222"
```

Single QMgr test

./imaclnt.sh -d 0 -tx 1:0:t:5:50000:2:0 -r 10000 -s 32-32 -wr -rrs

Two QMgr test

./imaclnt.sh -d 0 -tx 1:0:t:10:50000:2:0 -r 10000 -s 32-32 -wr -rrs

Three QMgr test

./imaclnt.sh -d 0 -tx 1:0:t:15:50000:2:0 -r 10000 -s 32-32 -wr -rrs

Four QMgr test

./imaclnt.sh -d 0 -tx 1:0:t:20:50000:2:0 -r 10000 -s 32-32 -wr -rrs

From *mqttbench* command line increase aggregate rate to target rate.

Machine 2, 3, 4, and 5

Start the WMQ QMgr

```
./crtqms.sh 1 1 50000 0-4
```

Machine 2

Start mqbench

```
export IMAClient=MQ ; export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf true -jb
QM1 -jt mqb -d ima2mq -db 1 -dx 5 -rl 0 -nt 5 -du true -cc 200 -tx
```

Machine 3

Start mqbench

```
export IMAClient=MQ ; export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf true -jb
QM1 -jt mqb -d ima2mq -db 6 -dx 10 -rl 0 -nt 5 -du true -cc 200 -tx
```

Machine 4

Start mqbench

```
export IMAClient=MQ ; export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf true -jb
QM1 -jt mqb -d ima2mq -db 11 -dx 15 -rl 0 -nt 5 -du true -cc 200 -tx
```

Machine 5

Start mqbench

```
export IMAClient=MQ ; export LargeConn=0
numactl --physcpubind=6-11 --membind=1 ./imaclnt.sh -tc Subscriber -jf true -jb
QM1 -jt mqb -d ima2mq -db 16 -dx 20 -rl 0 -nt 5 -du true -cc 200 -tx
```

FANIN.SEC

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256
export CERTSIZE=2048
export SSLClientMeth=TLSv12
export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904 16903"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -tx 0:0:/t/t:1:200000:2:0 -r 100000 -s 32-32 -wr
```

From *mqttbench* command line increase aggregate rate to 400K msgs/sec

Machine 2

```
export IMAClient=MQTT
export BatchingDelay=0
export UseSecureConn=1
export SSLCipher=AES256
export CERTSIZE=2048
export SSLClientMeth=TLSv12
export IMAPort="16903"
export IMAServer="10.10.0.222"

./imaclnt.sh -d 0 -rx 0:1:/t/+:1:1:1:0
```

FANOUT.SEC

Machine 1

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256
export CERTSIZE=2048
export SSLClientMeth=TLSv12
export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222
10.12.8.222"

./imaclnt.sh -d 0 -tx 0:0:t:10:5:2:0 -r 100000 -s 32-32 -wr -rrs
```

From *mqttbench* command line increase aggregate rate until appliance can no longer handle the incoming message rate.

Machine 2

```
export IMAClient=MQTT
export LargeConn=0
export DelayCount=1
export DelayTime=1
export BatchingDelay=0
export UseSecureConn=1
export SSLCipher=AES256
export CERTSIZE=2048
export SSLClientMeth=TLSv12
export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904 16903"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -rx 0:1:t:10:50:2:0 -rrs
```

LATENCY

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -tx 0:0:t:10:50000:2:0 -r 100000 -rrs -s 32-32 -wr
```

Machine 2

```
export IMAClient=MQTT
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 0 -rx 0:1:t:1:10:1:0
```

Machine 3

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=0
export SampleRate=10000
export IMAPort="16901 16902"
export IMAServer="10.10.0.222 10.12.0.222"

./imaclnt.sh -d 310 -b 10 -tx 0:0:lat:1:1:1:0 -rx 0:1:lat:1:1:1:0 -wr -
s 32-32 -r 10000 -T 0x81 -lcsv latency.csv
```

CONNBURST

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902 16901"
export IMAServer="10.10.0.222 10.12.0.222 10.10.2.222 10.12.2.222
10.10.4.222 10.12.4.222 10.10.6.222 10.12.6.222 10.10.8.222 10.12.8.222
10.10.10.222 10.12.10.222 10.10.12.222 10.12.12.222 10.10.14.222
10.12.14.222 10.12.16.222"

./imaclnt.sh -d 90 -rx 0:1:t:1:500000:2:0 -T 0xA0 -u 1.0e-6 -lcsv
connlatency.csv
```