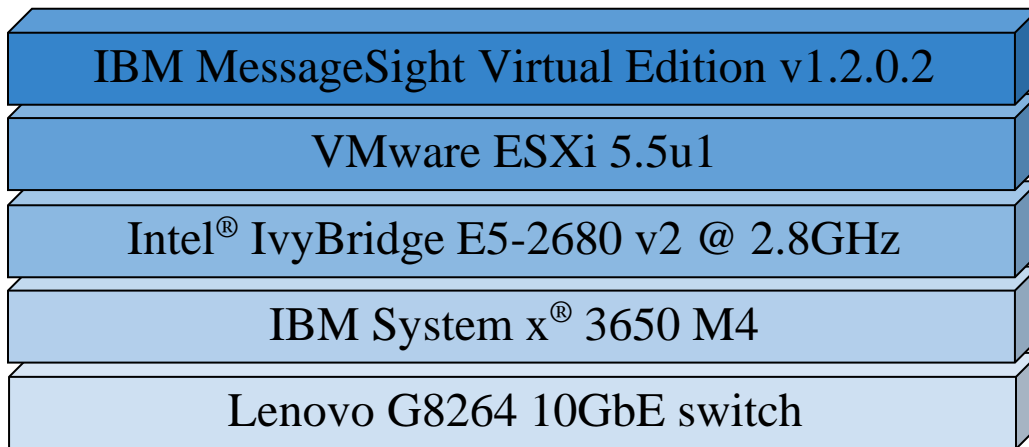


IBM MessageSight Virtual Edition v1.2.0.2 Performance Benchmarks

Tested by: IBM
Test date: July 31, 2015



Key Results

- IBM MessageSight Virtual Edition can scale up at a ratio of 4K concurrent connections per Gigabyte of memory allocated to the virtual machine.
- On a 16 core virtual machine with 128GB of memory
 - IBM MessageSight Virtual Edition is capable of processing in excess of 500K messages per second in the FANIN.Q0.PARTITIONS benchmark.
 - IBM MessageSight Virtual Edition can achieve a peak egress message rate of 100K msgs/sec to 512K MQTT consumer devices in the BROADCAST benchmark.

© Copyright IBM Corp. 2015 All Rights Reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Disclaimers

International Business Machines Corporation has prepared this report for your internal use only. It may not be redistributed, retransmitted, or published in any form without the prior written consent of IBM. All trademarks in this document belong to their respective owners, as further set forth on the following page.

The test results in this report for informational purposes only. IBM does not guarantee similar performance results. To the fullest extent permitted by applicable law without possibility of contractual waiver, all information contained herein is provided on an “AS-IS” BASIS, WITHOUT WARRANTY OF ANY KIND.

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

As used in this publication:

- 1) The following are trademarks or registered trademarks of International Business Machines Corporation, and registered in many jurisdictions worldwide:
 - IBM® and ibm.com®
 - MessageSight™
 - System x®
 - WebSphere®
- 2) Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries;
- 3) Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates;
- 4) Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both;

and
- 5) other company, product, or service names may be trademarks or service marks of others

TABLE OF CONTENTS

Disclaimers..... 2

Trademarks and Service marks 3

Benchmark specifications..... 5

 Overview 5

 Test case descriptions 6

Product background 33

Benchmark participants 34

Contacts 34

Methodology 34

 Test harness 34

 Limitations..... 35

System under test 36

 IBM MessageSight Virtual Edition 36

 VMware ESXi servers 36

 Network elements (switches)..... 36

 Client machines 36

 Client operating system and JVM..... 37

VMware ESXi Servers and MessageSight Virtual Machine (VM) configuration 38

 Performance considerations when choosing a VMware data store 38

 Configuring vSwitches and mapping virtual network interfaces..... 38

 Performance considerations when collocating a MessageSight VM on an ESXi Server with other VMs
 40

 Disabling Power Management features 42

MessageSight VM sizing..... 43

 Memory 43

 CPU 44

 Disk I/O 44

 Network I/O..... 44

 MessageSight VM sizes used in these benchmarks 45

Score card 46

 QoS 0 - Throughput..... 47

 QoS 1 - Throughput..... 48

 QoS 2 - Throughput..... 49

 HA - Throughput 50

 Connection Rates..... 51

Additional test results data 52

Appendix A: IBM MessageSight configuration 72

 Network interface configuration..... 72

 Messaging Hub configuration 72

 Security profile configuration..... 72

 Certificate profile configuration 72

Appendix B: Client machine tuning 73

 Kernel tuning parameters..... 73

 NIC tuning parameters..... 73

 User and process limits..... 74

 OS services stopped..... 74

 Interrupt and application CPU affinity tuning parameters..... 75

 JVM tuning parameters 77

Appendix C: Test harness execution details 78

Benchmark specifications

Overview

The performance benchmarks in this report test the ability of the IBM MessageSight Virtual Edition to handle high volumes of messaging traffic under a variety of different workloads, configurations, and messaging patterns. The workloads tested in these benchmarks reflect application architectures and messaging patterns which are commonly found in large scale networks of mobile and telemetry devices. The key performance metrics highlighted in this report are non-persistent messaging throughput, connection rate, and concurrent connection scaling.

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* and *JMSBenchTest*) 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 under controlled laboratory conditions.

Test case 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.

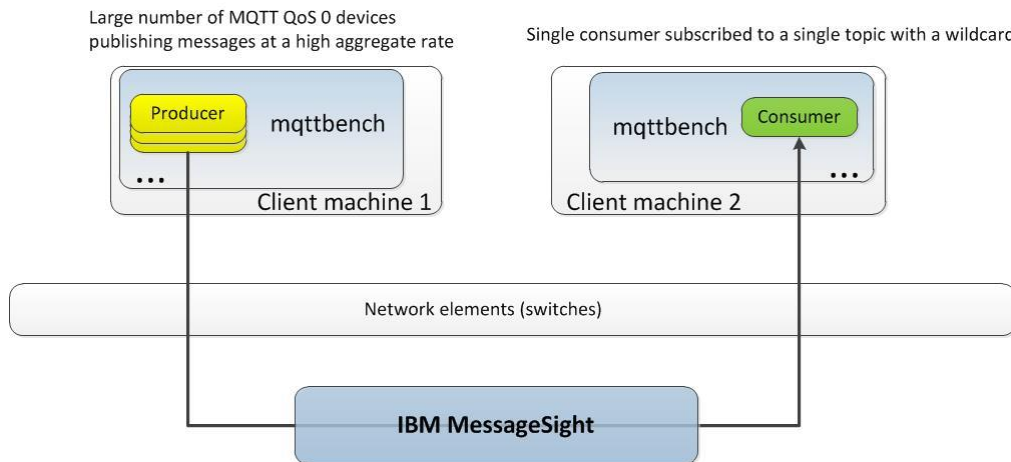
The messages used in the majority of the test cases described below are small messages, less than 512 bytes. The message sizes used in these test cases are intended to be representative of application data, such as GPS coordinates, sensor data, and other compact data that may be sent and received from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

The notation: **Q[x]** in the title represents one the 3 levels of MQTT “Quality of Service” (QoS). Due to time limitations not all combinations of QoS were performed for each test case.

FANIN.Q0.SINGLE

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a single MQTT v3.1.1 consumer.

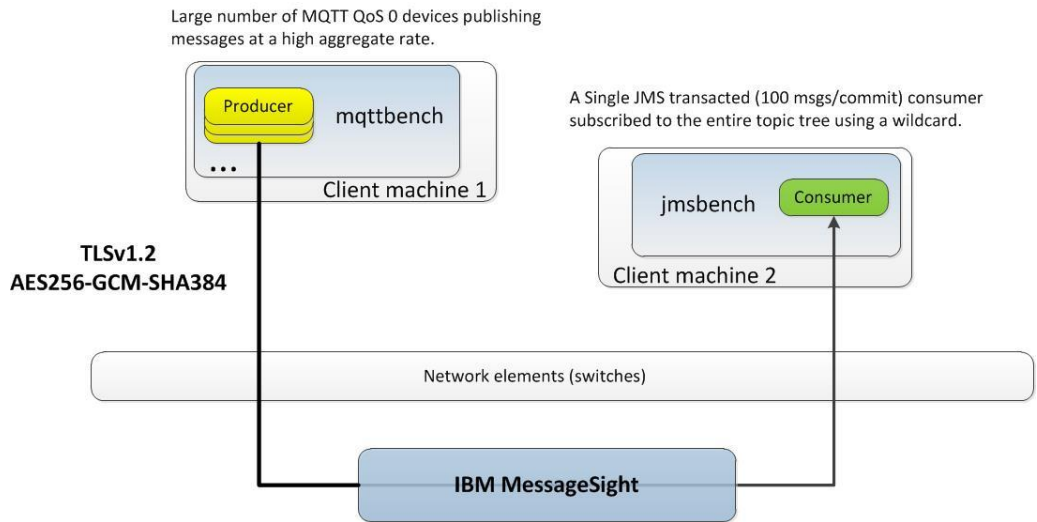
Each MQTT QoS 0 producer publishes messages on its own unique topic. The consumer subscribes to all topics using a wildcard. On one client machine a single instance of *mqttbench* is started which creates a consumer in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.SEC.Q0.JMS.SINGLE

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 QoS 0 producers publishing messages at a high aggregate rate over a secure connection to MessageSight, which then delivers the messages to a single JMS consumer. The *JMSBenchTest* benchmark application creates the JMS consumer using the MessageSight JMS client library.

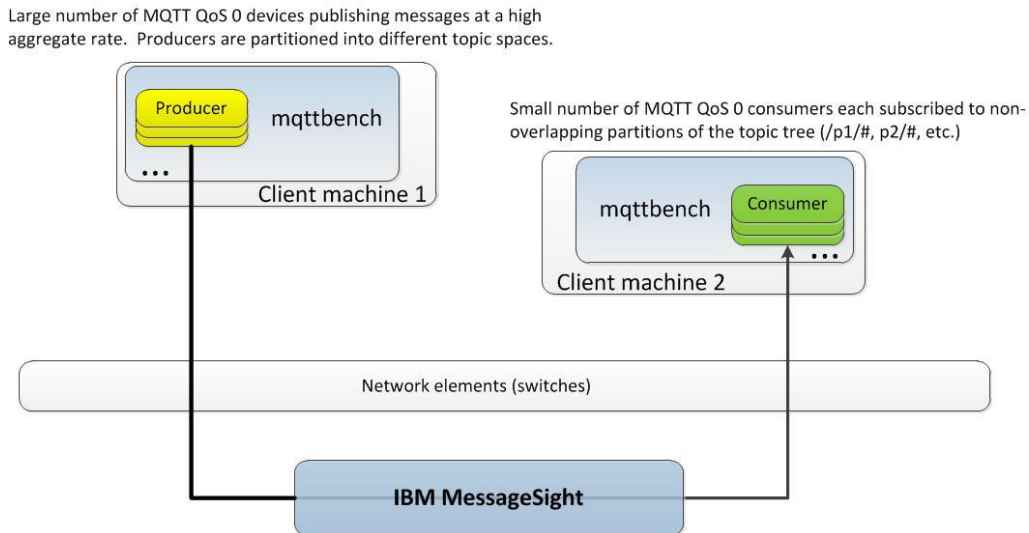
The MQTT producers connect via secured endpoints. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. Each producer publishes messages on its own unique topic. The consumer will subscribe to all topics using a wildcard. In this test case, the lowest quality of service messaging (“fire-and-forget”) allowed by the 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 MessageSight JMS client library message cache. On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates a single JMS connection, session, and consumer.



FANIN.Q0.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a small number of MQTT v3.1.1 consumers.

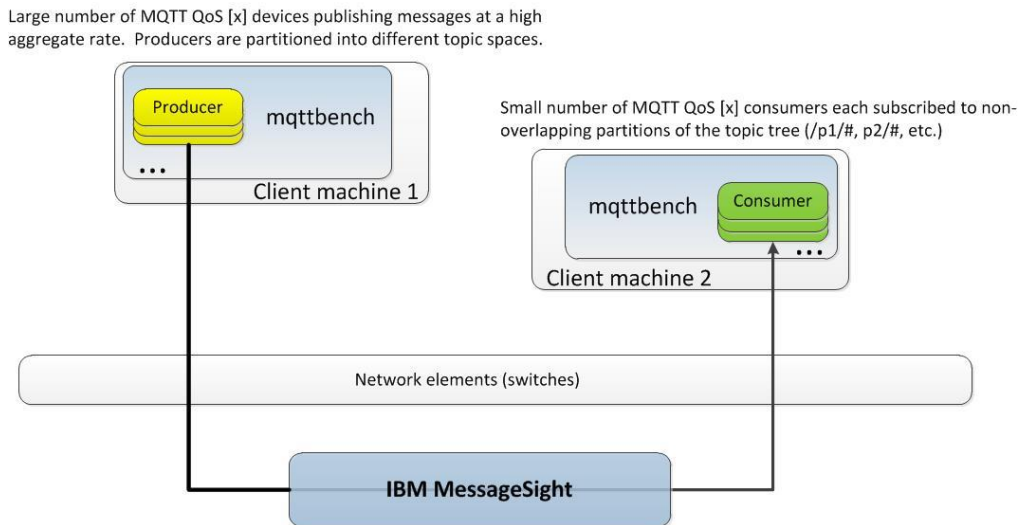
Each MQTT QoS 0 producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each MQTT QoS 0 consumer will subscribe to, and receive messages from, a single partition, and each partition is subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. For persistent messaging, the MQTT consumers connect with *cleansession=0* and subscribe to the topic. On one client machine a single instance of *mqttbench* is started which creates all consumers in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.Q[1|2].PERSISTENT.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a small number of MQTT v3.1.1 consumers.

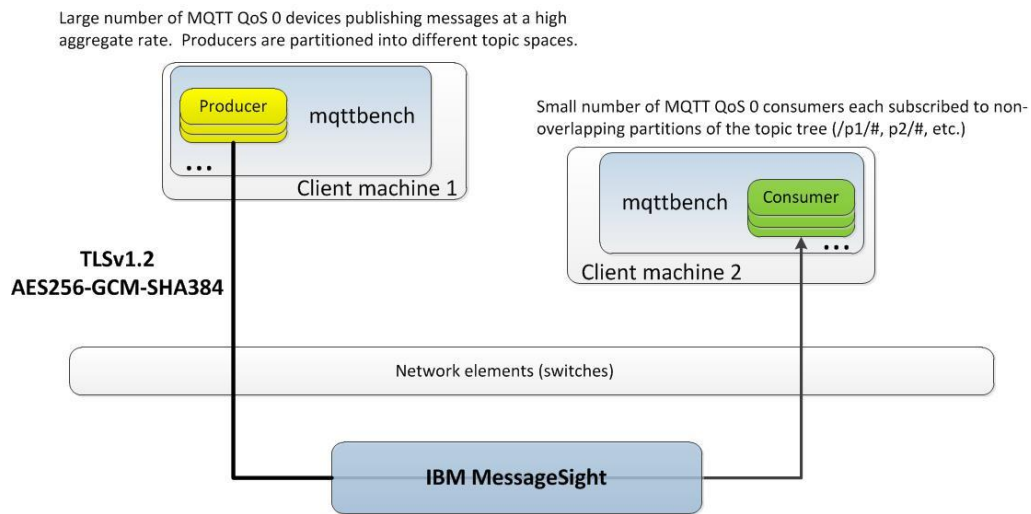
The messages are published/subscribed using QoS level 1 or level 2 and are persistent. Each MQTT (QoS x) producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each MQTT (QoS x) consumer will subscribe to, and receive messages from, a single partition, and each partition is subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. For persistent messaging, the MQTT consumers connect with *cleansession=0* and subscribe to the topic with QoS Level 1 or 2 (matching the producer). On one client machine a single instance of *mqttbench* is started which creates all consumers in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.SEC.Q0.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.Q0.PARTITIONS, except that, in this test, communications between the MQTT v3.1.1 producers and MessageSight are secured.

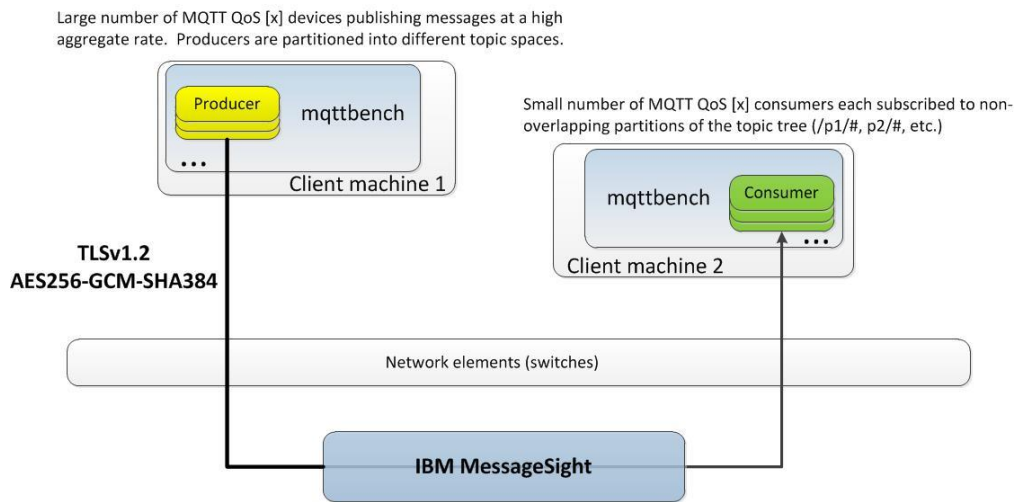
The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).



FANIN.SEC.Q[1|2].PERSISTENT.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.Q[1|2].PERSISTENT.PARTITIONS respectively, except that, in this test, communications between the MQTT v3.1.1 producers and MessageSight are secured.

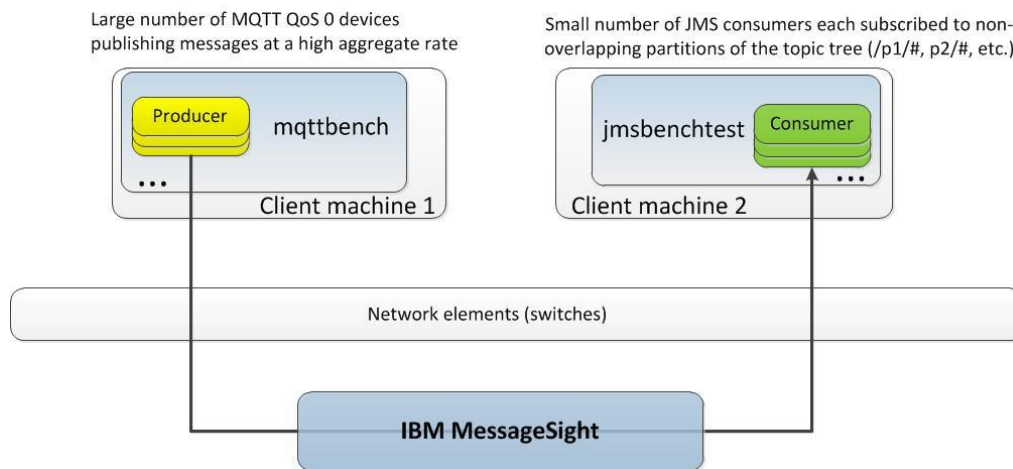
The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).



FANIN.Q0.JMS.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a small number of JMS consumers. The *JMSBenchTest* benchmark application creates the JMS consumers using the MessageSight JMS client library.

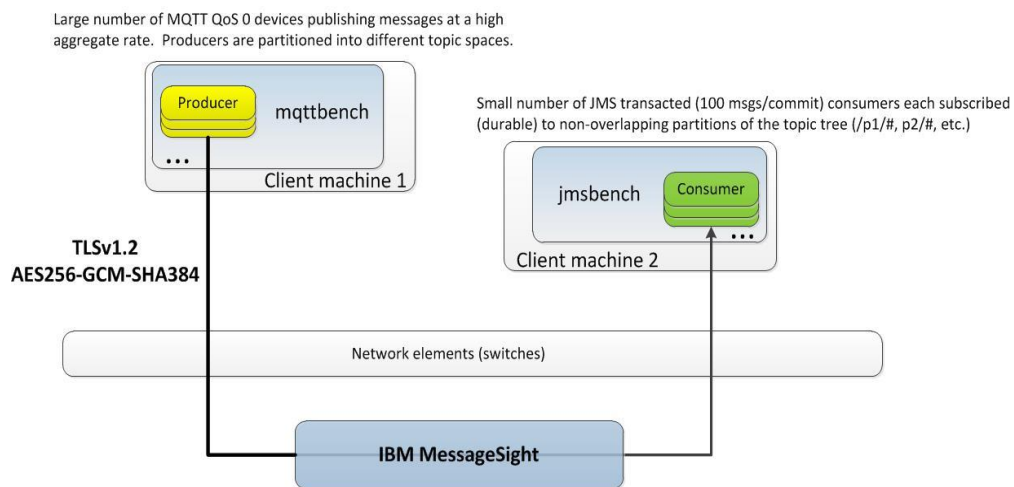
Each MQTT (QoS 0) producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each JMS consumer will subscribe to, and receive messages from, a single partition, and each partition is subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. In this test case, the lowest quality of service messaging (“fire-and-forget”) allowed by MessageSight JMS client library is used. Fire-and-forget messaging is equivalent to MQTT QoS 0: non-persistent, non-transactional, and ACKing is disabled. On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, sessions, and consumers which each subscribe to a different partition. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.SEC.Q0.JMS.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.Q0.JMS.PARTITIONS, except that, in this test, communications between the MQTT v3.1.1 producers and MessageSight are secured.

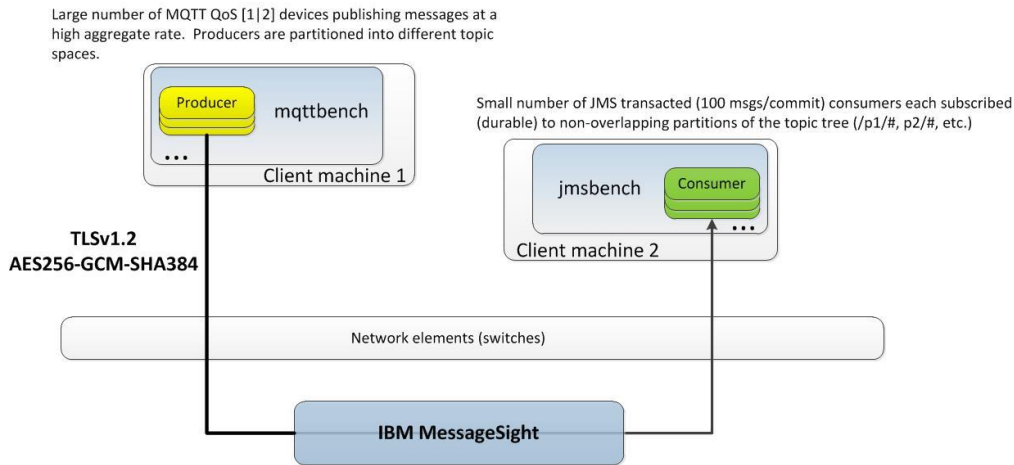
The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).



FANIN.SEC.Q[1|2].PERSISTENT.JMS.PARTITIONS

This test case measures maximum throughput rate. The workload in these test cases is identical to FANIN.SEC.Q0.JMS.PARTITIONS, except that, in this test, the messaging is persistent. Messages are stored to non-volatile media, which ensures consumers will receive messages missed, while disconnected, upon reconnecting. The MQTT producers will send QoS Level 1 messages in one test and QoS Level 2 messages in the other test.

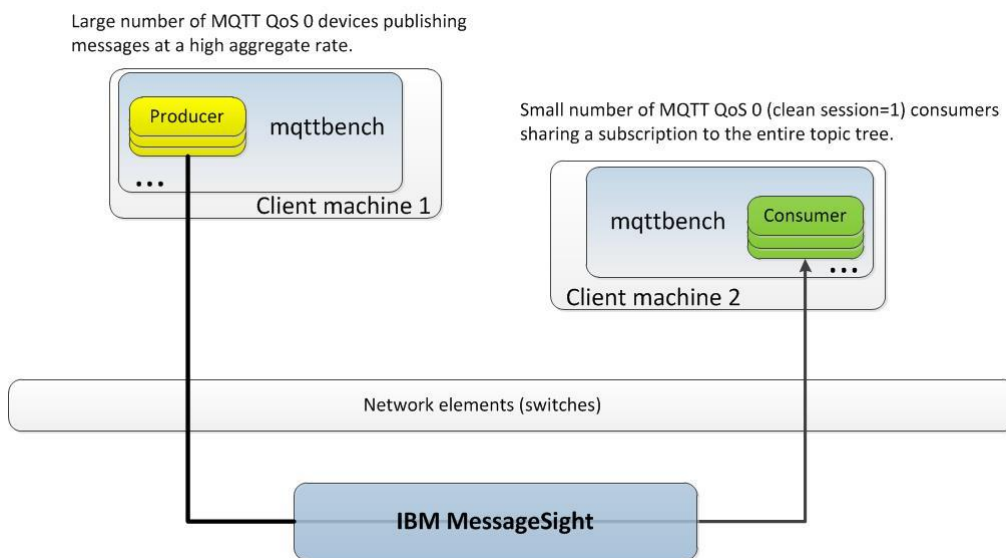
On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, transacted sessions and consumers. The ACK mode used by the JMS consumers is client application ACK mode and the consumer applications send ACKs at a rate of 100 received messages/ACK. Each QoS level will be tested with both 10 and 100 JMS consumers. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the MQTT producer load.



FANIN.Q0.SHARED

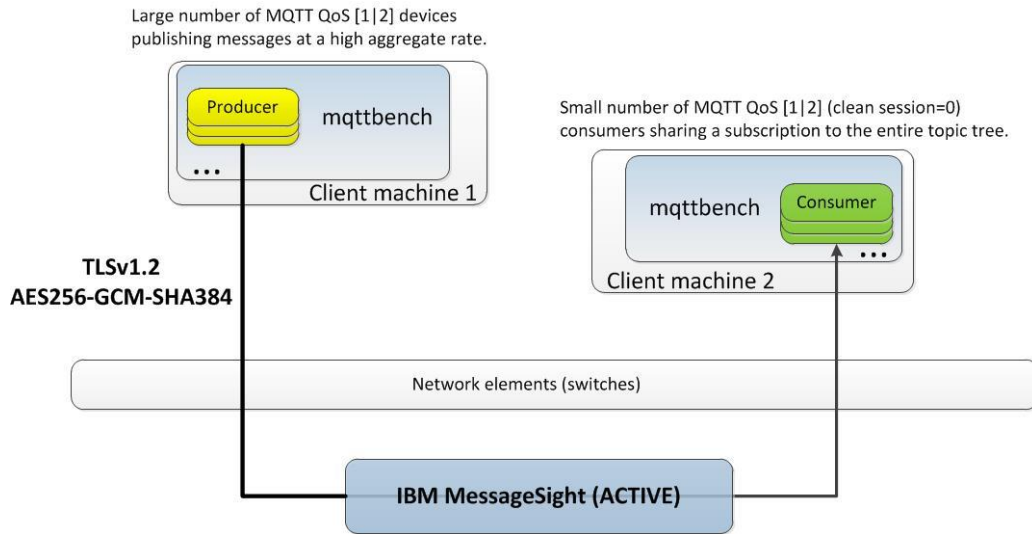
This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a small number of MQTT v3.1.1 consumers.

Each MQTT QoS 0 producer publishes messages on its own unique topic. The consumers share a subscription (“SharedSubscription”) to the entire topic tree. Each message from the subscription is delivered to only one of the shared subscribed consumers. On one client machine a single instance of *mqttbench* is started which creates a consumer in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.Q[1|2].PERSISTENT.SHARED

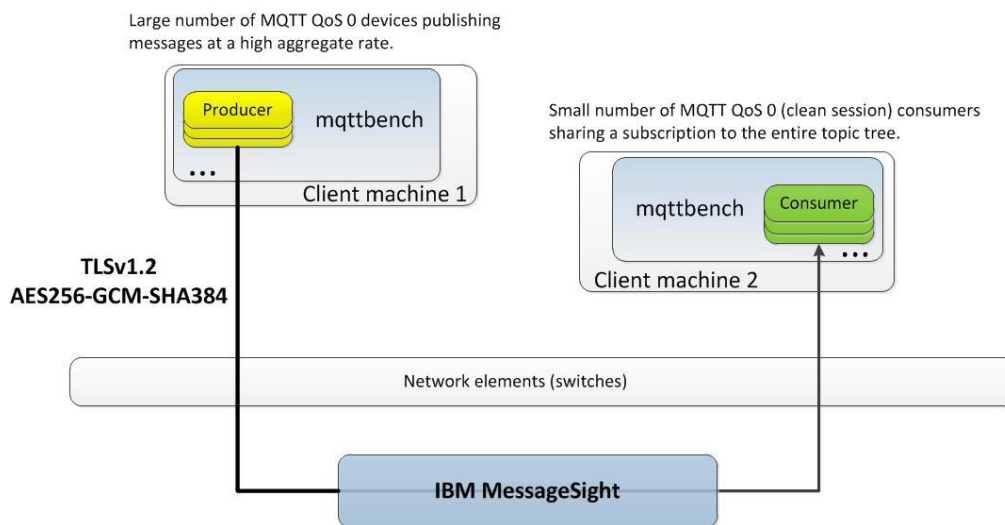
This test case measures maximum throughput rate. The workload in these test cases is identical to FANIN.Q0.SHARED, except that, in this test, persistent messaging is used. Persistent messaging ensures consumers receive messages missed, while disconnect, upon reconnecting via storing to disk. The MQTT producers will send QoS level 1 messages in one test and QoS level 2 messages in the other test. The MQTT consumers connect with the *cleansession* =0, and subscribe to the topic tree using the same level of QoS messaging as the producers.



FANIN.SEC.Q0.SHARED

This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.Q0.SHARED, except that, in these tests, communications between the MQTT v3.1.1 producers and MessageSight are secured.

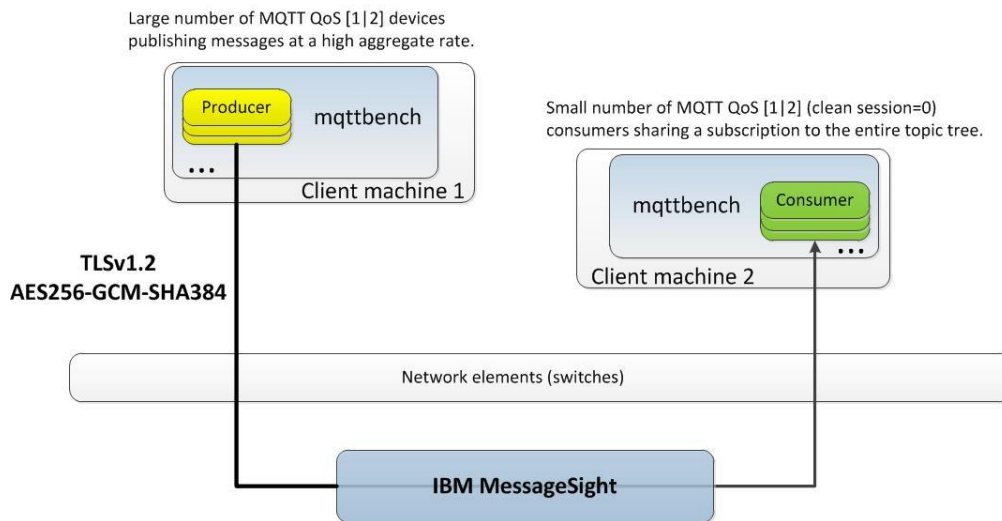
The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).



FANIN.SEC.Q[1|2].PERSISTENT.SHARED

This test case measures maximum throughput rate. The workload in these test cases are identical to FANIN.Q[1|2].PERSISTENT.SHARED, except that, in these tests, communications between the MQTT v3.1.1 producers and MessageSight are secured.

The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).

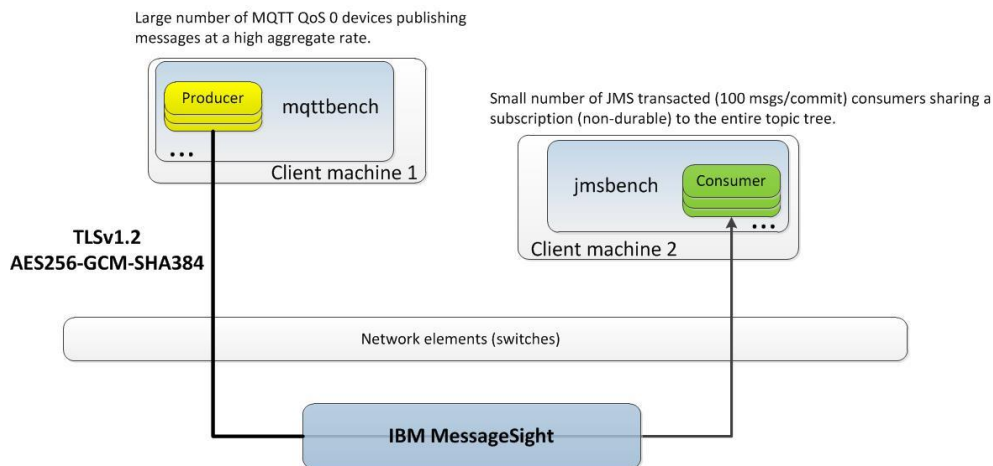


FANIN.SEC.Q0.JMS.SHARED

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight using secured communications, which then delivers the messages to a small number of JMS consumers using a shared subscription.

The MQTT producers connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).

Each MQTT (QoS x) producer publishes messages on its own unique topic. The *JMSBenchTest* benchmark application creates the JMS consumers using the MessageSight JMS client library. The JMS consumers share a subscription to the entire topic tree. Each message from the subscription is delivered to only one of the shared subscribed consumers. On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, sessions, and consumers which each subscribe to the same “shared” topic. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the producer load.

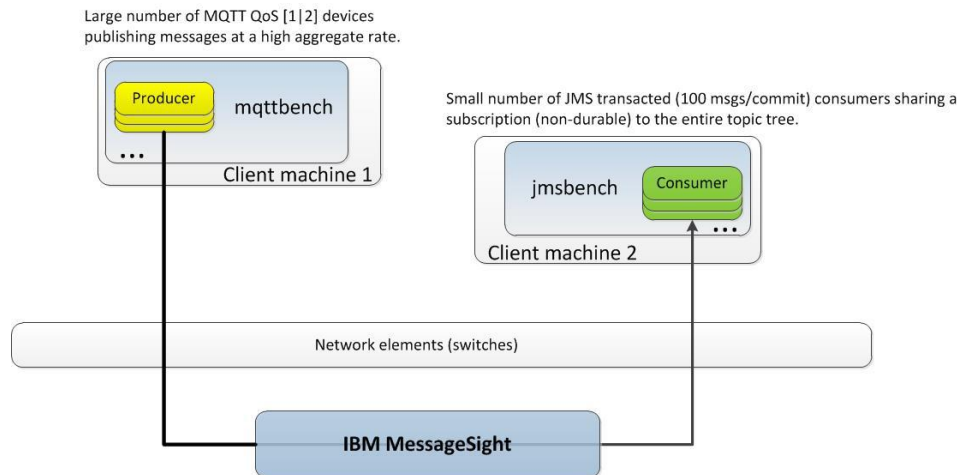


FANIN.Q[1|2].PERSISTENT.JMS.SHARED

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at a high aggregate rate to MessageSight, which then delivers the messages to a small number of JMS consumers.

The *JMSBenchTest* benchmark application creates the JMS consumers using the MessageSight JMS client library. Each MQTT (QoS x) producer publishes messages on its own unique topic. The JMS consumers share a durable subscription to the entire topic tree. Each message from the subscription is delivered to only one of the shared subscribed consumers. This test leverages the MessageSight JMS client library.

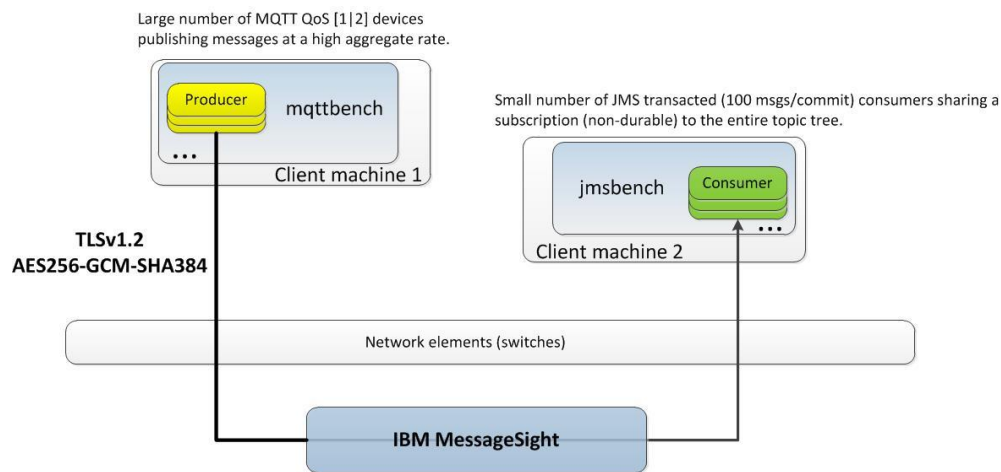
On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, transacted sessions and consumers. The ACK mode used by the JMS consumers is client application ACK mode and the consumer applications send ACKs at a rate of 100 received messages/ACK. Each QoS level will be tested with both 10 and 100 JMS consumers. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the MQTT producer load.



FANIN.SEC.Q[1|2].PERSISTENT.JMS.SHARED

This test case measures maximum throughput rate. The workload in these test cases are identical to FANIN.Q[1|2].PERSISTENT.JMS.SHARED, except that, in these tests, communications between the MQTT v3.1.1 producers and MessageSight are secured. The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#).

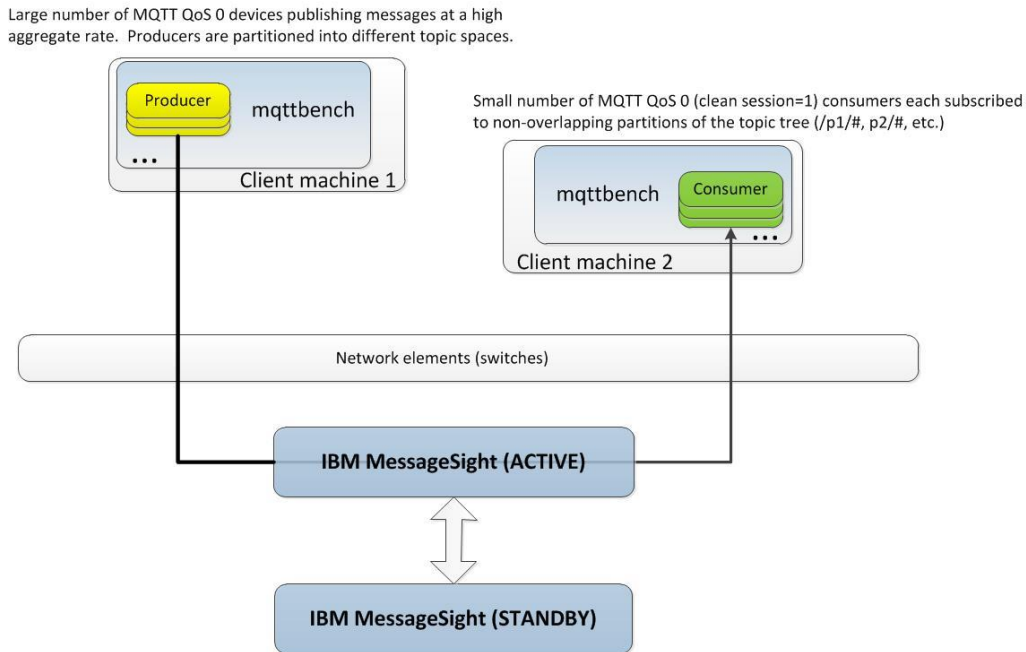
On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, transacted sessions and consumers. The ACK mode used by the JMS consumers is client application ACK mode and the consumer applications send ACKs at a rate of 100 received messages/ACK. Each QoS level will be tested with both 10 and 100 JMS consumers. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the MQTT producer load.



FANIN.Q0.HA.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at an aggregate rate to MessageSight, which then delivers the messages to a small number (1st – 10 and 2nd - 100) of MQTT v3.1.1 consumers. In this workload a pair of highly available MessageSight servers are used. The workload is targeted at the active node.

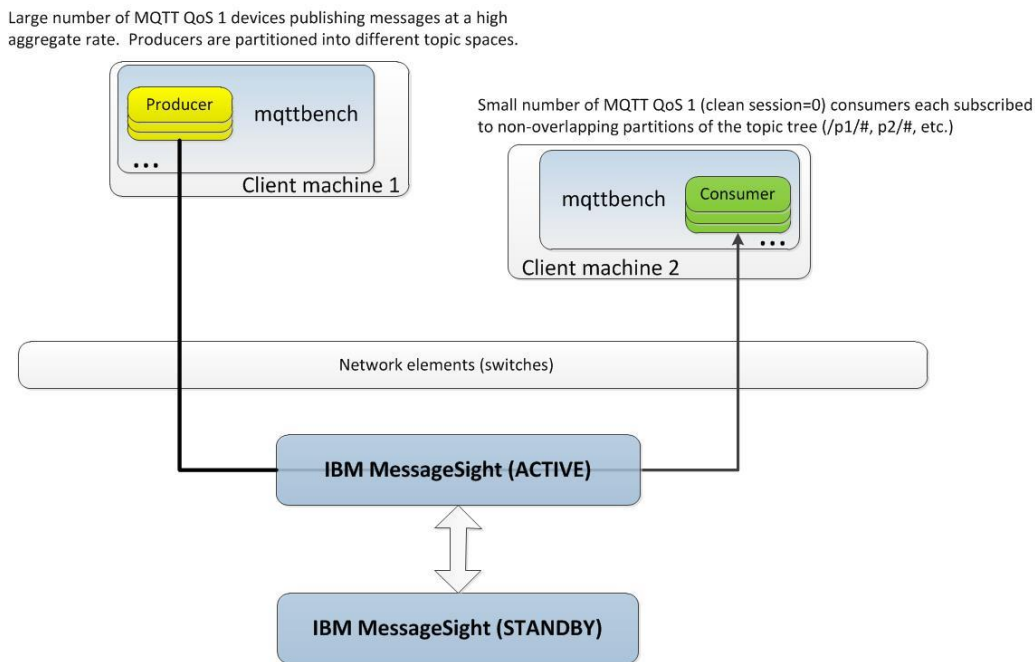
Each MQTT QoS 0 producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each MQTT (QoS 0) consumer will subscribe to, and receive messages from, a single partition, and each partition is subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. The MQTT consumers perform a MQTT Connect with *cleansession=1* and subscribing to the topic with QoS Level 0. On one client machine a single instance of *mqttbench* is started which creates all consumers in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.Q1.HA.PERSISTENT.PARTITIONS

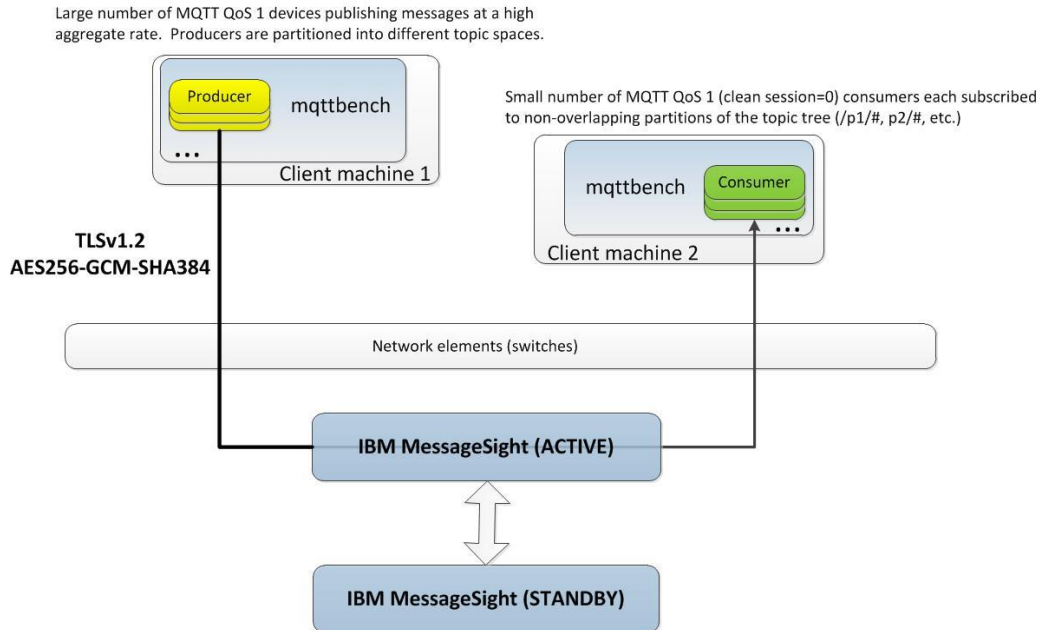
This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at an aggregate rate to MessageSight, which then delivers the messages to a small number (1st – 10 and 2nd - 100) of MQTT v3.1.1 consumers with persistent messaging enabled. In this workload a pair of highly available MessageSight servers are used. The workload is targeted at the active node.

Each MQTT QoS 1 producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each MQTT (QoS 1) consumer will subscribe to, and receive messages from, a single partition, and each partition is subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. Persistent messaging is requested via the MQTT consumers performing a MQTT Connect with *cleansession=0* and subscribing to the topic with QoS Level 1. On one client machine a single instance of *mqttbench* is started which creates all consumers in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS

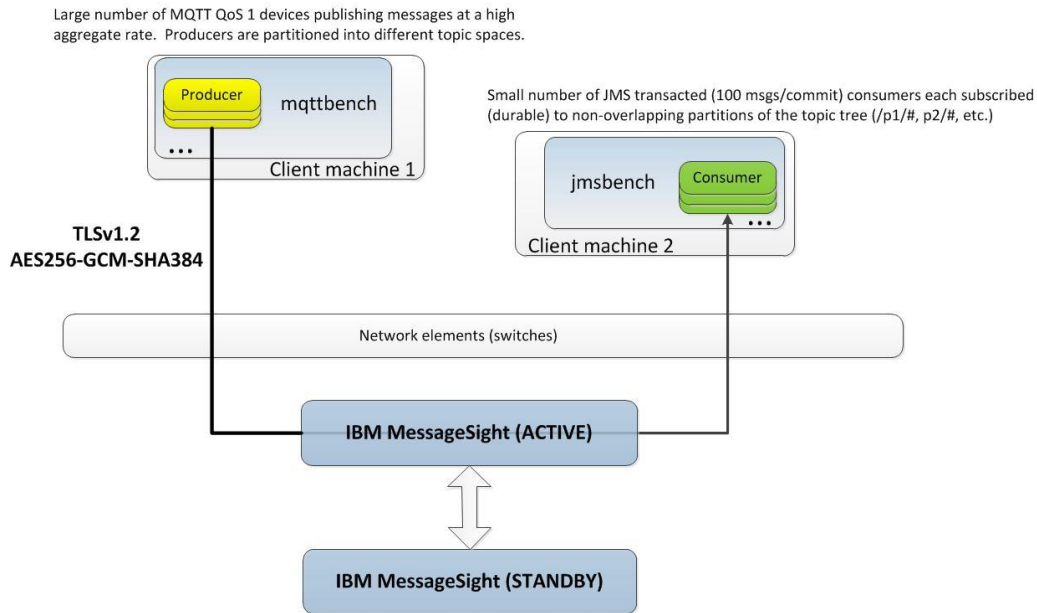
This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.Q1.HA.PERSISTENT.PARTITION, except that, in this test, communications between the MQTT v3.1.1 producers and MessageSight are secured. The MQTT producers in this test case connect to MessageSight over secure endpoints configured. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits.



FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS

This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.SEC.Q1.HA.PERSISTENT.PARTITION, except that, in this test, the MessageSight delivers the messages to a small number (1st - 10 and 2nd - 100) of JMS consumers.

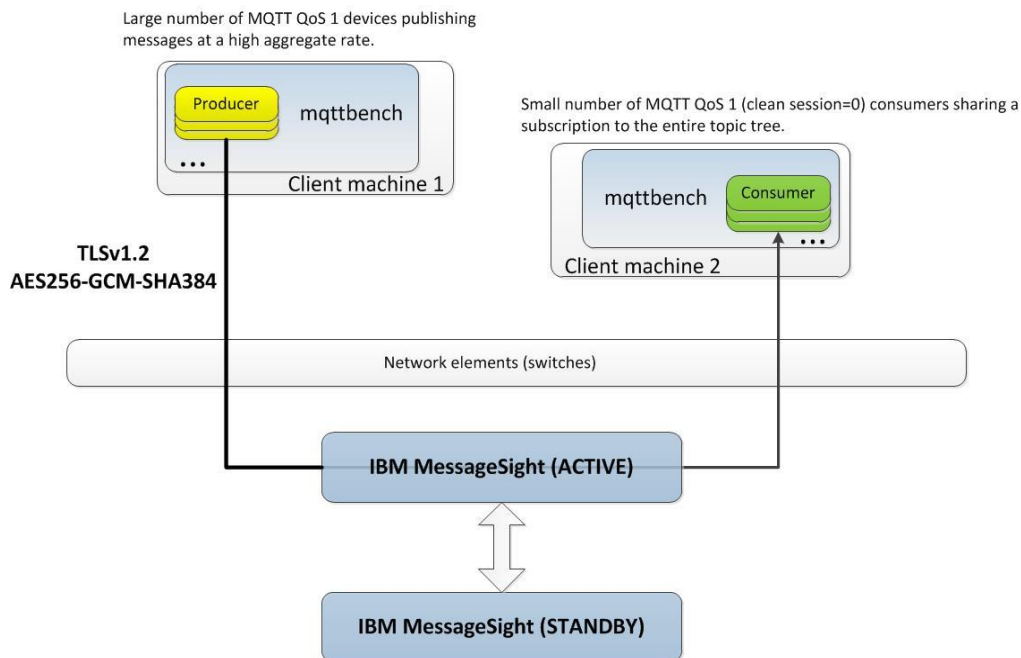
This test leverages the MessageSight JMS client library message cache. On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, transacted sessions and consumers. The ACK mode used by the JMS consumers is client application ACK mode and the consumer applications send ACKs at a rate of 100 received messages/ACK. The JMS consumers use a client message cache = 100K . Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the MQTT producer load.



FANIN.SEC.Q1.HA.PERSISTENT.SHARED

This test case measures maximum throughput rate. The workload in this test case is generated by a large number of MQTT v3.1.1 producers publishing messages at an aggregate rate to MessageSight using secured communications, which then delivers the messages to a small number of MQTT v3.1.1 consumers with persistent messaging enabled using a shared subscription. In this workload a pair of highly available MessageSight servers are used. The workload is targeted at the active node.

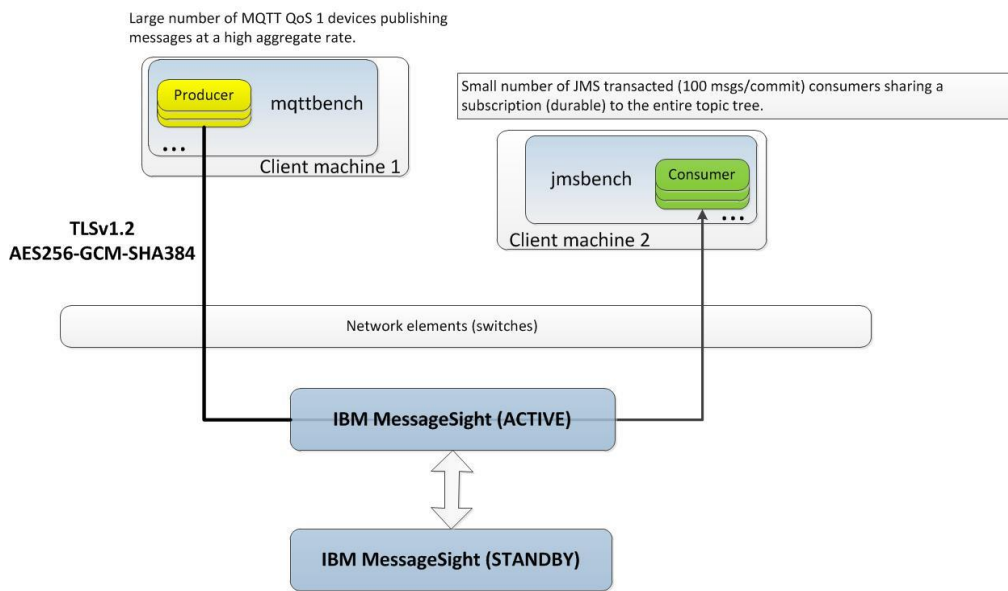
The MQTT producers connect via secured endpoints. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. Each MQTT QoS 1 producer publishes messages on its own unique topic. The topic tree is divided into non-overlapping partitions such that each MQTT (QoS 1) consumer will subscribe to, and receive messages from, a single partition, and all partitions are subscribed to by a single consumer. Subscriptions are constructed with a partition name followed by a wildcard. Persistent messaging is requested during the MQTT consumers perform a MQTT Connect by setting *cleansession=0* and subscribing to a topic with a “SharedSubscription” with QoS Level 1. On one client machine a single instance of *mqttbench* is started which creates all consumers in the test, subsequently a second instance of *mqttbench* on a second client machine is started and begins generating the producer load.



FANIN.SEC.Q1.HA.PERSISTENT.JMS.SHARED

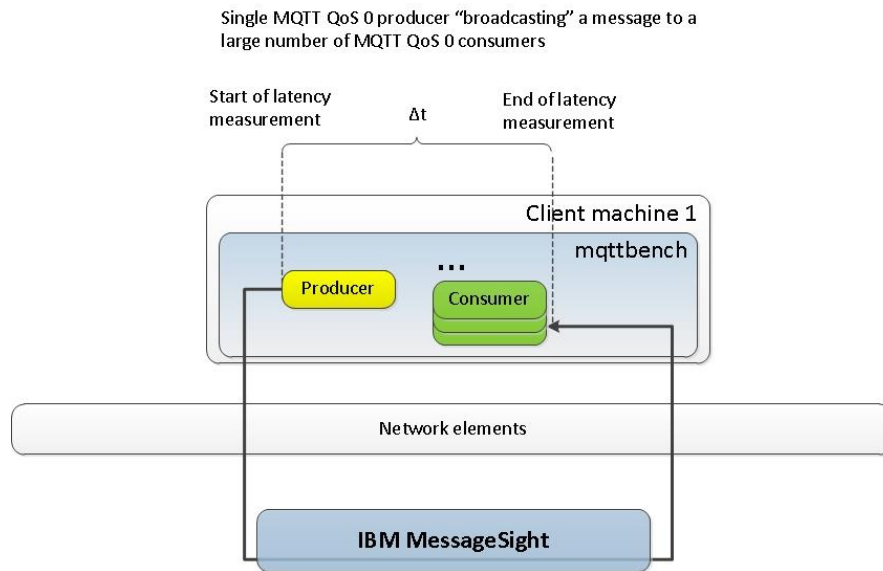
This test case measures maximum throughput rate. The workload in this test case is identical to FANIN.SEC.Q1.HA.PERSISTENT.SHARED, except that, in this test, the MessageSight delivers the messages to a small number (1st – 10 and 2nd – 100) of JMS consumers.

On one of the client machines a single JVM is started. From this JVM, *JMSBenchTest* is started and creates multiple JMS connections, transacted sessions and consumers. 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/ACK. The JMS consumers use a client message cache = 100K and a durable shared subscription. Subsequently, an instance of *mqttbench* on a second client machine is started and begins generating the MQTT producer load.



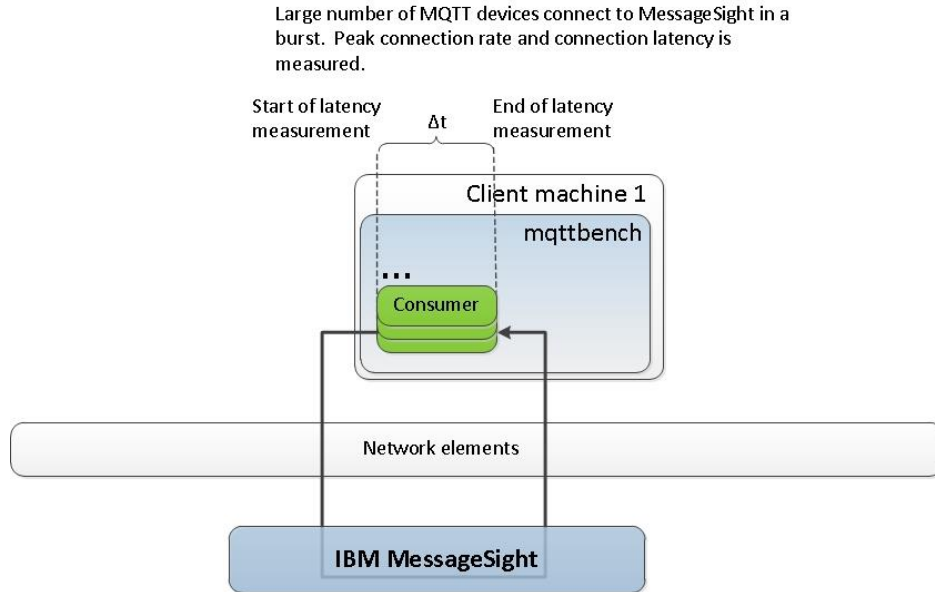
BROADCAST

This test case measures both peak egress message rate and message latency. The workload in this test case is generated by a single MQTT v3.1.1 QoS 0 producer publishing 1 message every 20 seconds to MessageSight, which then delivers the message to a large number of MQTT v3.1.1 QoS 0 consumers all subscribed to the “broadcast” topic. On one client machine a single instance of *mqttbench* is started which creates all consumers in the test and the single producer. Each message published by the producer is timestamped by referencing the local TSC clock. The same TSC is referenced again for each consumer receiving the message. Message latency is then calculated from the receive time and send time. Peak egress message rate is captured from the MessageSight statistics. Message latency includes network latency and client latency



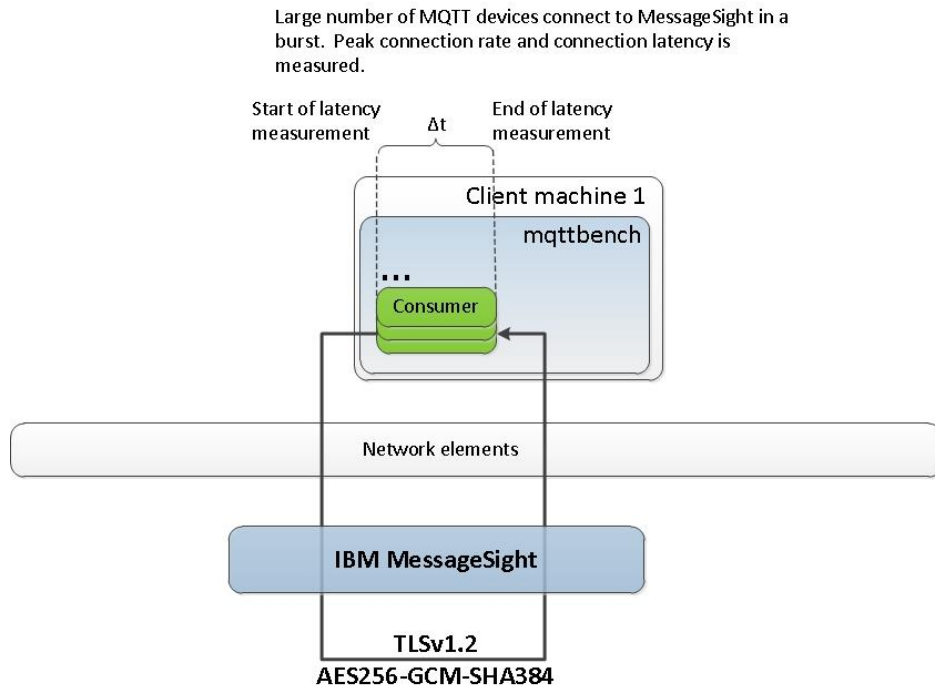
CONNBURST

This test case measures the peak connection rate that can be sustained by MessageSight and connection latency. In this workload a large number of MQTT v3.1.1 devices connect to MessageSight using non-secure communications. On client machine 1, a single instance of *mqttbench* starts the MQTT devices which connect to the appliance. The total time to connect all MQTT devices is measured and the average connection rate is calculated from this measurement. Connection latency includes network latency and client latency.



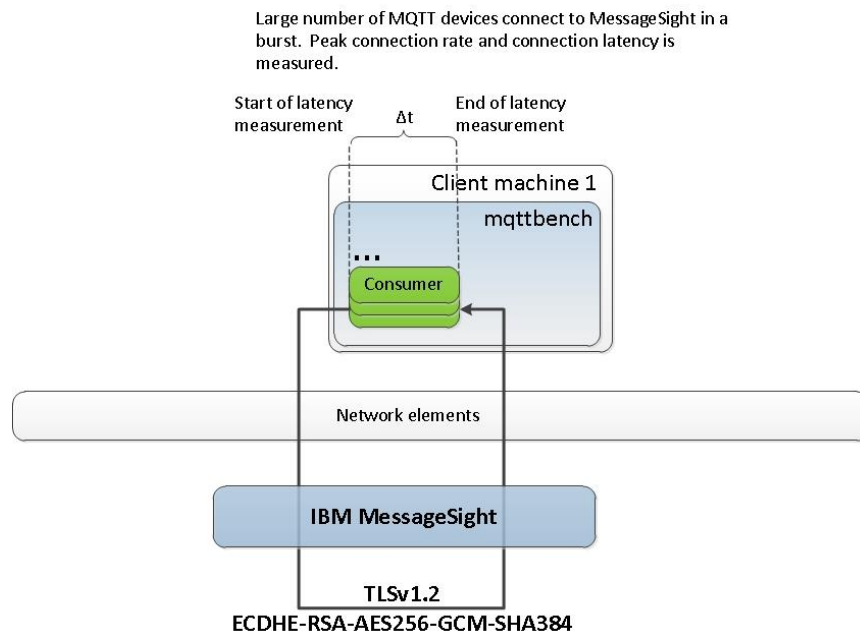
CONNBURST.SEC

This test case is identical to CONNBURST, except that communications are secured. In this test case two secure endpoints configured on MessageSight are used for secure communications. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is AES256-GCM-SHA384. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#). On client machine 1, a single instance of *mqttbench* starts the MQTT devices which connect to MessageSight. Connection latency includes network latency and client latency.



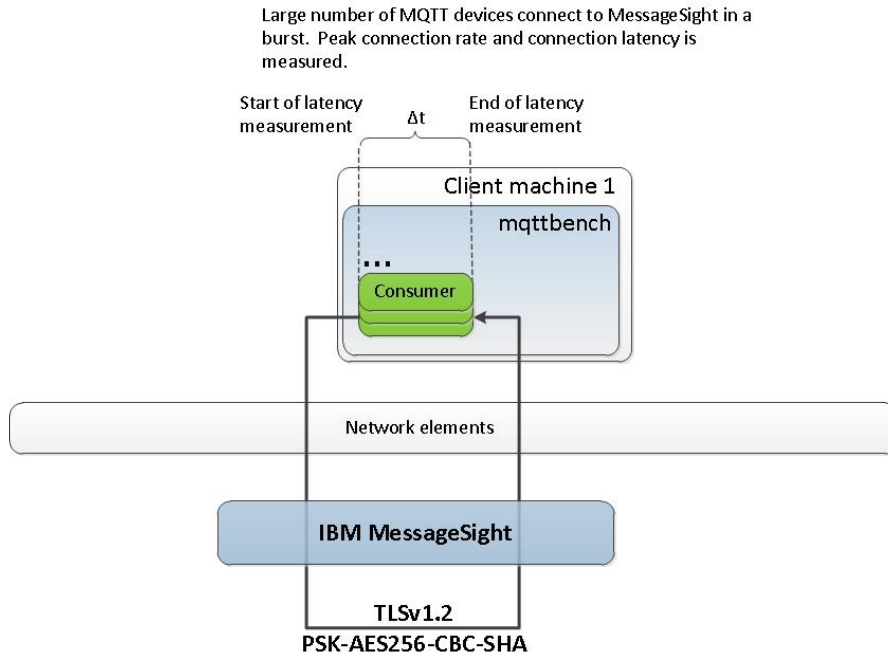
CONNBURST.SEC.ECDHE

This test case is identical to CONNBURST, except that communications are secured. In this test case two secure endpoints configured on MessageSight are used for secure communications. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is ECDHE-RSA-AES256-GCM-SHA384. ECDHE, or elliptic curve Diffie Hellman key exchange provides forward secrecy to improve protection against MITM attacks, but come at a cost to performance. The server key used in this test case is 2048 bits. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#). On client machine 1, a single instance of *mqttbench* starts the MQTT devices which connect to MessageSight. Connection latency includes network latency and client latency.



CONNBURST.SEC.PSK

This test case is identical to CONNBURST, except that communications are secured. In this test case two secure endpoints configured on MessageSight are used for secure communications. The TLS protocol used in this test case is TLSv1.2, the cipher category configured in the MessageSight security profile is “BEST” and the negotiated cipher is PSK-AES256-CBC-SHA. PSK, or pre-shared keys based ciphers are frequently used by resource constrained embedded devices. The pre-shared key used in this test case is 16 bytes. For more detailed information on how the secure endpoints are configured on MessageSight see [Appendix A: IBM MessageSight configuration](#). On client machine 1, a single instance of *mqttbench* starts the MQTT devices which connect to MessageSight. Connection latency includes network latency and client latency.



Product background

IBM MessageSight Virtual Edition:

MessageSight Virtual Edition allows you to deploy MessageSight to virtual data centers. With VMware ESXi, you can rapidly deploy one or more MessageSight virtual appliances to provide device connectivity.

Designed to be able to sit at the edge of the internet, 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. 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 MessageSight ideal to deliver large amounts of data to analytics engines and other big data types of application.

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

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

Lenovo G8264 10GbE switch:

The Lenovo System Networking RackSwitch G8264 is a 10/40 Gigabit Ethernet (GbE) aggregation switch designed for the data center, providing speed, intelligence and interoperability on a proven platform.

The Lenovo RackSwitch G8264 offers up to 48x10GbE ports and 4x40 GbE ports, which can also be used as a high-density 10 GbE switch, with 1.28 Tbps—in a 1U footprint. The G8264 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.

Designed with top performance in mind, the Lenovo RackSwitch G8264 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 (HA) features enable the G8264 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 Virtual Edition
- 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 in the [Test case descriptions](#) section above.

Throughput tests must remain stable for five minutes in order to be considered as passing. During throughput tests subscription queues are monitored for number of buffered messages and for buffered messages high water marks (HWM). The number of buffered messages can fluctuate, but must not be monotonically increasing. Throughput tests that result in monotonically increasing buffered messages are considered failing tests.

Latency tests run for five minutes over which times several hundreds of thousands of latency measurements are recorded. Latency statistics are calculated from this sample population.

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	<i>mqttbench v1.0</i>
JMS workload generator	<i>JMSBenchTest v1.0</i> <i>MessageSight JMS client library v1.2</i>
Clock source (for latency measurements)	<i>TSC</i>

The *mqttbench* test harness program:

- A multi-threaded C program that can simulate up to 1M MQTT v3.1.1 devices
- mqttbench does NOT use the open source Eclipse Paho MQTT client, in particular mqttbench will batch multiple messages per network read/write on high throughput connections. For QoS 1 and higher the inflight message window (unACKed messages) is configurable in mqttbench and defaults to 64K unACKed messages.
- 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 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

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 MessageSight 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 MessageSight, 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 Virtual Edition

MessageSight version	1.2.0.2
MessageSight build ID	20150708-0800

VMware ESXi servers

VMware ESXi version	5.5 update 1
Vendor and model	IBM System x 3650 M4 (Model #: 7915AC1)
Processors	2 x Intel Xeon 10-core E5-2680 v2 2.80 GHz
L3 cache	25MB
Memory speed	1600 MHz
Memory size	128GB
RAID	IBM ServeRAID M5110e with 1GB Write Cache
Disks	RAID 10 with 4 x 900GB 10K rpm 2.5" SAS drives
PCIe bus speed	PCIe Gen3 x8 lanes
NIC	Intel x520 dual port 10GbE Intel I350 quad port 1GbE
NIC Driver version	igxbe 3.19.1.iov igb 5.0.5.1
BIOS version	IBM uEFI 1.50 (Build ID: VVE134OUS)

Network elements (switches)

Vendor and model	Lenovo G8264 10GbE RackSwitch (Rear to Front airflow) Layer 2/Layer 3 switch
Software version	Networking OS v7.4.1
Hardware revision	0
Rack units	1U
Power supplies	2 x 450W
Interop guide	ibm.com/networking
Cables	IBM 3m 10GbE DAC SFP+

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	128GB
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

VMware ESXi Servers and MessageSight Virtual Machine (VM) configuration

The VMware ESXi host configuration and the MessageSight VM settings used in these benchmarks are described in this section. For benchmarks which leverage MessageSight HA (not to be confused with VMware based HA) two identical ESXi servers were used. For details on the hardware and software specifications of the ESXi servers used in these benchmarks see the [VMware ESXi servers](#) section above. One of the ESXi servers was selected to host the PRIMARY MessageSight VM and the other ESXi server was selected to host the STANDBY MessageSight VM.

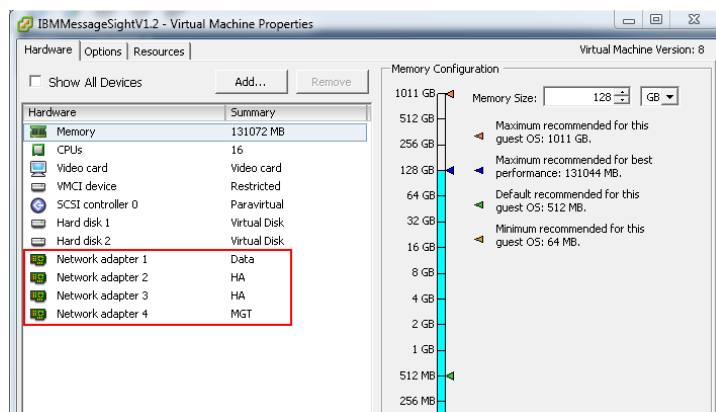
Performance considerations when choosing a VMware data store

When planning a MessageSight Virtual Edition deployment it is important to give consideration to the disk subsystem on which the MessageSight Virtual Edition is deployed. For persistent messaging workloads it is particularly important to deploy *Hard disk 2* (i.e. the persistent messaging store disk) of the MessageSight VM on a fast disk subsystem. Persistent messaging workloads will write to the store disk frequently and disk I/O latency becomes important. For this reason an ESXi server with a fast disk subsystem should be chosen to deploy MessageSight Virtual Edition.

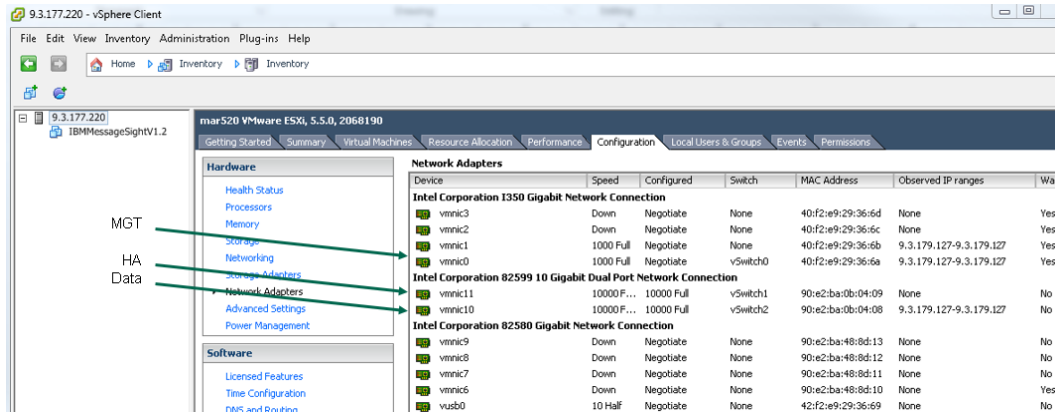
In these benchmarks the ESXi servers have the IBM ServeRAID M5110e RAID controller with a 1GB write cache. The ESXi server local datastore on which the MessageSight VM is deployed is a RAID 10 array consisting of 4 x 900GB 10K rpm 2.5” SAS drives.

Configuring vSwitches and mapping virtual network interfaces

The MessageSight VM is configured with 4 virtual network interfaces. One network interface is dedicated for management traffic, one for messaging traffic (i.e. Data), one for HA data replication, and one of HA discovery (see the screenshot of the MessageSight VM properties below).



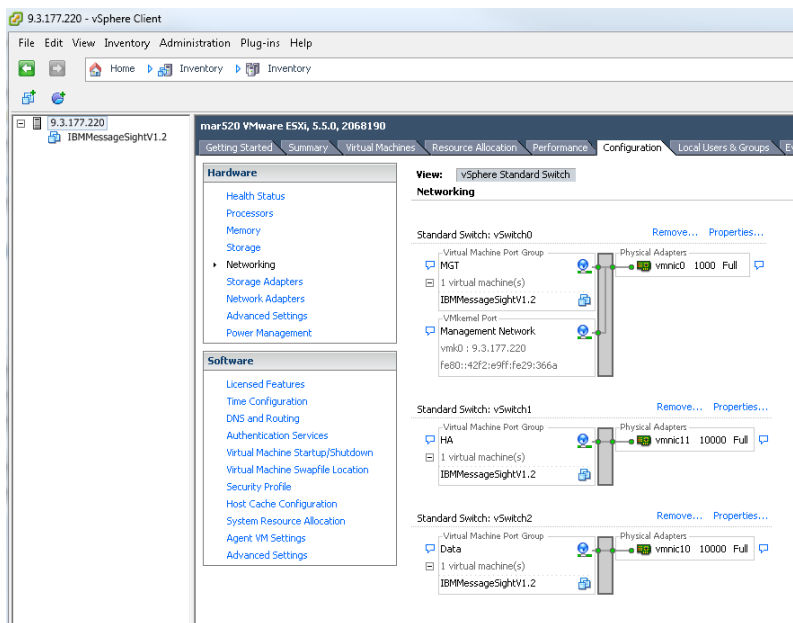
In these benchmarks the ESXi servers have both Intel 1GbE and 10GbE network interfaces.



The ESXi servers were configured with three vSphere Standard Switches: vSwitch0, vSwitch1, and vSwitch2. vSwitch0 (alias MGT) maps to the physical host network interface named vmnic0, which is a 1GbE network interface. vSwitch1 (alias HA) maps to the physical host network interface named vmnic11, which is a 10GbE network interface. Finally, vSwitch2 (alias Data) maps to the physical host network interface named vmnic10, which is also a 10GbE network interface.

The virtual network interfaces on the MessageSight VM map to the vSwitches as follows:

VM properties	Guest OS device	VMware vSwitch
Network adapter 1	eth1	vSwitch2 (alias Data)
Network adapter 2	ha0	vSwitch1 (alias HA)
Network adapter 3	ha1	vSwitch1 (alias HA)
Network adapter 4	eth0	vSwitch0 (alias MGT)



It is recommended to separate the MessageSight management traffic from the HA and messaging traffic to prevent delayed response times for administrative commands. A 1GbE interface is adequate for management traffic. It is also recommended to separate messaging (i.e. Data) traffic from MessageSight HA traffic, for this reason messaging and HA traffic are mapped to different physical host network interfaces. In these benchmark tests two 10GbE interfaces were used for messaging and HA traffic, but the workload will determine the speed of the network interface. It is recommended to use 10GbE interfaces for messaging and HA traffic. It is not strictly required to map HA discovery and HA replication traffic on different host network interfaces, but doing so may improve resilience against heartbeat timeouts between members of an HA pair. In these benchmark tests it was determined that mapping both HA discovery and HA replication traffic to the same 10GbE host network interface did not result in any heartbeat timeouts. It is also reasonable to map the management traffic and HA discovery traffic to the same 1GbE host network interface, since these are both low throughput classes of traffic.

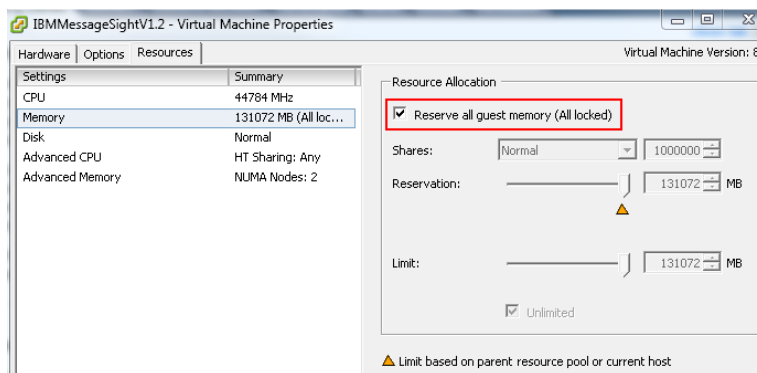
Performance considerations when collocating a MessageSight VM on an ESXi Server with other VMs

When planning a MessageSight Virtual Edition deployment it is important to consider the resources required by MessageSight as well as the resources required by the other VMs collocated on the same ESXi host server.

Memory resource

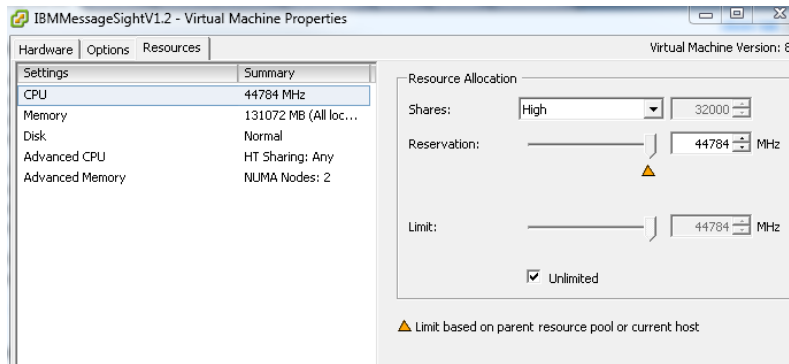
VMware ESXi servers use a memory management technique known as “ballooning” whereby the ESXi server may harvest “unused” memory from guest VMs when the host is under memory pressure. For more details on this memory management technique refer the following VMware article: http://www.vmware.com/files/pdf/perf-vsphere-memory_management.pdf

It is recommended to reserve (or lock) all guest memory on the MessageSight VM to prevent untimely harvesting of “unused” MessageSight VM memory by the ESXi host. Note, that ESXi server will not allow a VM to lock all of the memory on the ESXi server.



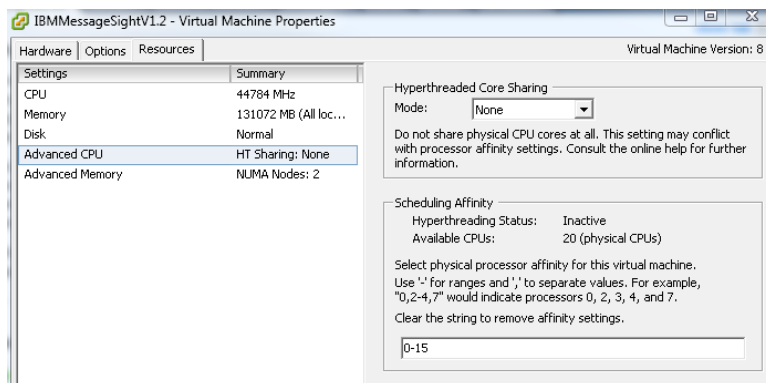
CPU resource

The same recommendation applies to the CPU resource. It is recommended to reserve (or lock) all guest CPU MHz allocated to the VM to prevent untimely CPU starvation of the MessageSight VM by other VMs running on the ESXi server. For example, on an ESXi server with 2.8GHz cores and a VM that is allocated 16 cores, in order to reserve all allocated CPU cycles that would be $(2799 * 16) \approx 44800$ MHz.



It is also recommended that when running multiple VMs on an ESXi server to isolate the MessageSight VM on a dedicated set of CPUs, away from other lower priority VMs by setting the CPU affinity of the other VMs on a non-overlapping CPU set. For example, on a dual socket Intel server, deploy the MessageSight VM on the same socket where the high speed network interfaces are installed and deploy all other lower priority VMs on the other CPU socket.

Note, before attempting to affinitize VMs to specific CPUs it is important to know how the logical CPUs in the ESXi host OS map to physical cores (or hyperthreads) on the processors. It is optimal to keep the entire VM on a single CPU socket if the number of CPUs allocated to the VM is less than or equal to the number of cores per CPU socket, in order to leverage the CPU cache as effectively as possible. CPU affinity tuning of VMs is considered an advanced tuning exercise and should not be attempted unless you have the prerequisite knowledge described above.



Datstore resource

As previously mentioned *Hard disk 2* (i.e. store disk) is used for persistent messaging. The store disk must be at least 4X the size of the memory allocated to the MessageSight VM.

For more detailed information on MessageSight configuration see Appendix A: IBM MessageSight configuration.

Disabling Power Management features

To prevent untimely CPU throttling it is recommended to disable power capping on VMware ESXi host CPUs and CPU frequency scaling features within the system BIOS/uEFI and VMware ESXi configuration, which can result in reduced performance in order to reduce power consumption.

MessageSight VM sizing

There are four critical resources to consider when planning a MessageSight Virtual Edition deployment: memory, CPU, disk I/O, and network I/O. Each resource impacts a different aspect of MessageSight performance. This section, along with the Score card section below, provide the guidance and data that will be needed in sizing exercises before planning a deployment of MessageSight Virtual Edition.

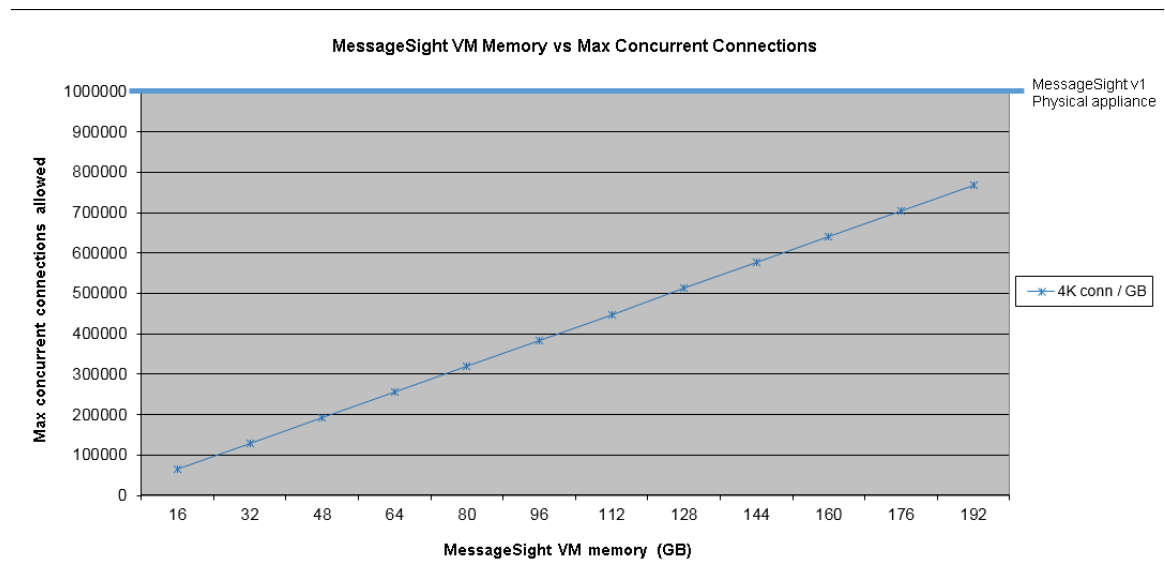
Memory

The memory allocated to the MessageSight VM affects several aspects of MessageSight scalability. The list below enumerates some of the key MessageSight resources affected by the amount of memory allocated to the MessageSight VM.

- concurrent device connections
- buffered messages
- retained messages
- subscriptions
- inflight transactions

As more memory is allocated to the MessageSight VM the number of the MessageSight resources above also increases. The minimum amount of memory that can be allocated to a MessageSight VM for production usage is 16GB.

The maximum number of concurrent connections allowed by MessageSight Virtual Edition is a function of the amount of memory allocated the MessageSight VM. MessageSight allows 4K concurrent device connections for each Gigabyte of memory allocated to the MessageSight VM. Below is a graphical representation of how the maximum number of concurrent device connections scales with the memory allocated to the MessageSight VM.



CPU

The number of CPUs allocated to the MessageSight VM affects several aspects of MessageSight performance. The list below enumerates some of the key MessageSight metrics affected by the number of CPUs allocated to the MessageSight VM.

- connection rate
- non-persistent, non-transactional messaging rate

As more CPUs are allocated to the MessageSight VM the performance of these metrics increases. See the [Score card](#) section below for test results of different CPU configurations allocated to the MessageSight VM. The minimum number of CPUs that can be allocated to a MessageSight VM for production usage is 4.

Disk I/O

Disk I/O latency, most notably disk write latency, of the disk subsystem used by the MessageSight store disk affects several aspects of MessageSight performance. The list below enumerates some of the key MessageSight metrics affected by the disk write latency of the MessageSight VM store disk.

- persistent messaging rate
- transactional messaging rate
- MessageSight HA replication rate

Network I/O

Network I/O is also important host resource which affects several aspects of MessageSight performance. Given that the nature of Internet of Things messaging and traffic patterns tends towards small packets at high rates, the underlying networking infrastructure must be able to handle several hundreds of thousands of packets per second. For this reason using high speed network interfaces such as 10GbE or 40GbE interfaces, which are SR-IOV capable, is critical to achieve high packet rates. The list below enumerates some of the key MessageSight metrics affected by the number of CPUs allocated to the MessageSight VM.

- connection rate
- non-persistent, non-transactional messaging rate
- MessageSight HA replication rate

MessageSight VM sizes used in these benchmarks

The table below provides detailed specifications of the three MessageSight VM sizes used in this benchmark testing.

Resource	Small	Medium	Large
CPU cores	8	12	16
Memory (GB)	32	64	128
Store Disk (GB)	128	256	512
Max device connections	128K	256K	512K

Although the largest memory configuration tested in these benchmarks is 128GB, MessageSight will scale the max number of concurrent connections beyond 128GB. Keep in mind, however, that a balance between the key resources listed above is important. For example, provisioning a VM with more memory may allow more concurrent connections, but if there is not sufficient CPU or Disk I/O resources to service those connections then they can experience delays.

The ratio of memory per core in these MessageSight VM configuration ranges from 4GB to 8GB per core, which is a reasonable ratio for most workloads. The ratio will vary based on the specific workload. For large connection/low throughput workloads a slightly higher ratio is recommended. For small connection/high throughput workloads a lower ratio is recommended.

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 the test case name alongside the test results. Detailed descriptions of each test case are provided in the [Test case descriptions](#) section above.

For each test case, results below are divided into 3 rows shown in order of the number of cores allocated to the MessageSight VM: 8, 12, and 16. The *VM size* column denotes the size of the VM next to each test result.

In addition to providing a summary of the benchmark the *Description* column also denotes the units in which the test results are reported. The *VALUE* column indicates the test result. For benchmarks in which message latency or connection latency are measured, the columns: AVG (average), 50P (50th percentile), 95P (95th percentile), 99P (99th percentile), and STDV (standard deviation) contain the latency statistics.

Type	Name	Description	VM Size	VALUE	AVG	50P	95P	99P	STDV
Type	Test case name	Additional test case description details	S	Results for 8 core VM					
			M	Results for 12 core VM					
			L	Results for 16 core VM					

Note that for *all* benchmarks, the number of device connections used in each test is based on the size of the MessageSight VM. For example, the FANIN.Q0.PARTITIONS benchmark on the 12 core VM with 64GB of memory involves 256K device connections sending messages, whereas the same benchmark on the 8 core VM with 32GB of memory involves only 128K device connections sending messages. The [Additional test results data](#) section below provides additional test result details and charts for those test cases which require additional test results data.

In the test cases that used secured MQTT producers for the connection to the MessageSight used TLSv1.2 and AES256-GCM-SHA384 cipher. These test cases are identified by “SEC” in the test case name.

QoS 0 - Throughput

The results shown below are with 100 consumers (except for the “SINGLE” workloads).

Type	Name	Description	VM Size	VALUE
Throughput	FANIN.Q0.SINGLE	Maximum throughput rate. A large number of MQTT QoS 0 producers sending messages to a 1 MQTT QoS 0 consumer. Message size 32 bytes. (msgs/sec)	S	140K
			M	140K
			L	140K
	FANIN.SEC.Q0.JMS.SINGLE	Maximum throughput rate. A large number of secured MQTT QoS 0 producers sending messages to a single JMS “QoS 0 equivalent” consumer. Message size 32 bytes. (msg/sec)	S	250K
			M	220K
			L	260K
	FANIN.Q0.SHARED	Maximum throughput rate. A large number of MQTT QoS 0 producers sending messages to a small number of MQTT QoS 0 consumers with shared subscription. Message size 32 bytes. (msgs/sec)	S	330K
			M	340K
			L	380K
	FANIN.Q0.PARTITIONS	Maximum throughput rate. A large number of MQTT QoS 0 producers sending messages to a small number of MQTT QoS 0 consumers. Message size 32 bytes. (msgs/sec)	S	420K
			M	450K
			L	470K
	FANIN.SEC.Q0.PARTITIONS	Maximum throughput rate. Identical to FANIN.Q0.PARTITIONS, but with producers connected to MessageSight using secured connection. Message size 32 bytes. (msgs/sec)	S	270K
			M	300K
			L	400K
	FANIN.Q0.JMS.PARTITIONS	Maximum throughput rate. A large number of MQTT QoS 0 producers sending messages to a small number of JMS “QoS 0 equivalent” consumers. Message size 32 bytes. (msgs/sec)	S	340K
			M	460K
			L	550K
	FANIN.SEC.Q0.JMS.PARTITIONS	Maximum throughput rate. Identical to FANIN.Q0.JMS.PARTITIONS, but with producers connected to MessageSight using TLSv1.2 and AES256-GCM-SHA384 cipher. Message size 32 bytes. (msgs/sec)	S	250K
			M	275K
			L	320K
	FANIN.SEC.Q0.JMS.SHARED	Maximum throughput rate. A large number of secured MQTT QoS 0 producers, sending messages to a small number of JMS consumers with shared subscription. Message size 32 bytes. (msgs/sec)	S	250K
			M	270K
			L	250K

QoS 1 - Throughput

The results shown below are with 100 consumers.

Type	Name	Description	VM Size	VALUE
Throughput	FANIN.Q1.PERSISTENT.PARTITIONS	Maximum throughput rate. Identical to FANIN.Q0.PARTITIONS, but sending/receiving MQTT QoS 1 messages. Message size 32 bytes. (msgs/sec)	S	38K
			M	40K
			L	45K
	FANIN.SEC.Q1.PERSISTENT.PARTITIONS	Maximum throughput rate. Identical to FANIN.Q1.PERSISTENT.PARTITIONS, but with producers connected to MessageSight using secured connection. Message size 32 bytes. (msgs/sec)	S	38K
			M	40K
			L	44K
	FANIN.SEC.Q1.PERSISTENT.JMS.PARTITIONS	Maximum throughput rate. Identical to FANIN.SEC.Q0.JMS.PARTITIONS, but sending/receiving MQTT QoS 1 persistent messages. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K
	FANIN.Q1.PERSISTENT.SHARED	Maximum throughput rate. A large number of MQTT QoS 1 producers sending messages to a small number of MQTT QoS 1 consumers with shared subscriptions. Message size 32 bytes. (msgs/sec)	S	40K
			M	40K
			L	44K
	FANIN.SEC.Q1.PERSISTENT.SHARED	Maximum throughput rate. Identical to FANIN.Q1.PERSISTENT.SHARED, but with producers connected to MessageSight using secured connections. Message size 32 bytes. (msgs/sec)	S	38K
			M	44K
			L	46K
	FANIN.Q1.PERSISTENT.JMS.SHARED	Maximum throughput rate. A large number of MQTT QoS 1 producers sending messages to a small number of JMS consumers with shared subscription. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K
	FANIN.SEC.Q1.PERSISTENT.JMS.SHARED	Maximum throughput rate. Identical to FANIN.Q1.PERSISTENT.JMS.SHARED, but with producers connected to MessageSight using secured connections. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K

QoS 2 - Throughput

The results shown below are with 100 consumers.

Type	Name	Description	VM Size	VALUE
Throughput	FANIN.SEC.Q2.PERSISTENT.PARTITIONS	Maximum throughput rate. Identical to FANIN.SEC.Q0.PARTITIONS, but sending/receiving MQTT QoS 2 messages. Message size 32 bytes. (msgs/sec)	S	26K
			M	30K
			L	34K
	FANIN.SEC.Q2.PERSISTENT.JMS.PARTITIONS	Maximum throughput rate. Identical to FANIN.SEC.Q0.JMS.PARTITIONS, but sending/receiving MQTT QoS 2 persistent messages. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K
	FANIN.Q2.PERSISTENT.SHARED	Maximum throughput rate. Identical to FANIN.Q1.PERSISTENT.SHARED but sending/receiving QoS 2 level messages. Message size 32 bytes. (msgs/sec)	S	28K
			M	32K
			L	33K
	FANIN.SEC.Q2.PERSISTENT.SHARED	Maximum throughput rate. Identical to FANIN.SEC.Q1.PERSISTENT.SHARED but sending/receiving QoS 2 level messages. Message size 32 bytes. (msgs/sec)	S	28K
			M	33K
			L	34K
	FANIN.Q2.PERSISTENT.JMS.SHARED	Maximum throughput rate. Identical to FANIN.SEC.Q1.JMS.PERSISTENT.SHARED but sending/receiving QoS 2 level messages. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K
	FANIN.SEC.Q2.PERSISTENT.JMS.SHARED	Maximum throughput rate. Identical to FANIN.Q2.JMS.PERSISTENT.SHARED but producers connected to MessageSight using secured connections. Message size 32 bytes. (msgs/sec)	S	9K
			M	9K
			L	9K

HA - Throughput

The results shown below are with 100 consumers.

Type	Name	Description	VM Size	VALUE
Throughput	FANIN.Q0.HA.PARTITIONS	Maximum throughput rate. A large number of MQTT QoS 0 producers sending messages to a small number of MQTT QoS 0 consumers. HA-pair, Message size 32 bytes. (msgs/sec)	S	390K
			M	480K
			L	490K
	FANIN.Q1.HA.PERSISTENT.PARTITIONS	Maximum throughput rate. A large number of MQTT QoS 1 producers sending messages to a small number of MQTT QoS 1 consumers. HA-pair, Message size 32 bytes. (msgs/sec)	S	18K
			M	20K
			L	25K
	FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS	Maximum throughput rate. Identical to FANIN.Q1.HA.PERSISTENT.PARTITIONS but producers connected to MessageSight using secured connections. Message size 32 bytes. (msgs/sec)	S	17K
			M	20K
			L	25K
	FANIN.SEC.Q1.HA.PERSISTENT.SHARED	Maximum throughput rate. A large number of secured MQTT QoS 1 producers sending messages to a small number of MQTT QoS 1 consumers with a shared subscription. HA-Pair, Message size 32 bytes. (msgs/sec)	S	20K
			M	27K
			L	37K
	FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS	Maximum throughput rate. A large number of secured MQTT QoS 1 producers sending persistent messages to a small number of JMS consumers. HA-Pair, Message size 32 bytes. (msgs/sec)	S	5K
			M	6K
			L	7K
	FANIN.SEC.Q1.HA.PERSISTENT.JMS.SHARED	Maximum throughput rate. A large number of secured MQTT QoS 1 producers sending persistent messages to a small number of JMS consumers with a shared subscription. HA-Pair, Message size, 32 bytes. (msgs/sec)	S	5K
			M	5K
			L	6K

Connection Rates

Type	Name	Description	VM Size	VALUE	AVG	50P	95P	99P	STDV
Connection rate	BROADCAST ¹	Peak egress message rate (msgs/sec) and message latency (milliseconds). Message size, 32 bytes.	S	25K					
			M	50K					
			L	100K					
	CONNBURST ²	Peak connection rate that can be sustained by IBM MessageSight (conn/sec) and connection latency (milliseconds). Connections are non-secure.	S	33.9K	0.7	0.4	0.7	0.8	14.8
			M	31.5K	0.8	0.4	0.6	1.4	15.9
			L	31.8K	0.9	0.4	0.6	1.3	17.2
	CONNBURST.SEC	Peak connection rate that can be sustained by IBM MessageSight (conn/sec) and connection latency (milliseconds). Connections use protocol TLSv1.2, cipher AES256-GCM-SHA384, and a 2Kb server key.	S	1.8K	9.8	8.1	22.2	36.7	7.5
			M	1.8K	3.7	3.2	6.2	9.3	1.7
			L	1.8K	3.4	2.9	4.6	7.9	3.6
	CONNBURST.SEC.ECDHE	Peak connection rate that can be sustained by IBM MessageSight (conn/sec) and connection latency (milliseconds). Connections use protocol TLSv1.2, cipher ECDHE-RSA-AES256-GCM-SHA384, and a 2Kb server key.	S	1.4K	2.7	1.0	14.4	33.4	6.2
			M	1.4K	6.0	5.5	9.2	14.0	2.0
			L	1.4K	5.4	4.9	7.2	11.4	3.9
	CONNBURST.SEC.PSK	Peak connection rate that can be sustained by IBM MessageSight (conn/sec) and connection latency (milliseconds). Connections use protocol TLSv1.2, cipher PSK-AES256-CBC-SHA, and a 16 byte pre-shared key.	S	1.4K	0.9	0.9	1.1	1.3	0.5
			M	1.4K	1.1	1.0	1.3	1.5	0.5
			L	1.4K	1.2	1.1	1.4	1.9	1.6

¹ The peak egress message rate and message latency are heavily affected by the number of consumer connections in the test. See the [Additional test results data](#) for the BROADCAST benchmark.

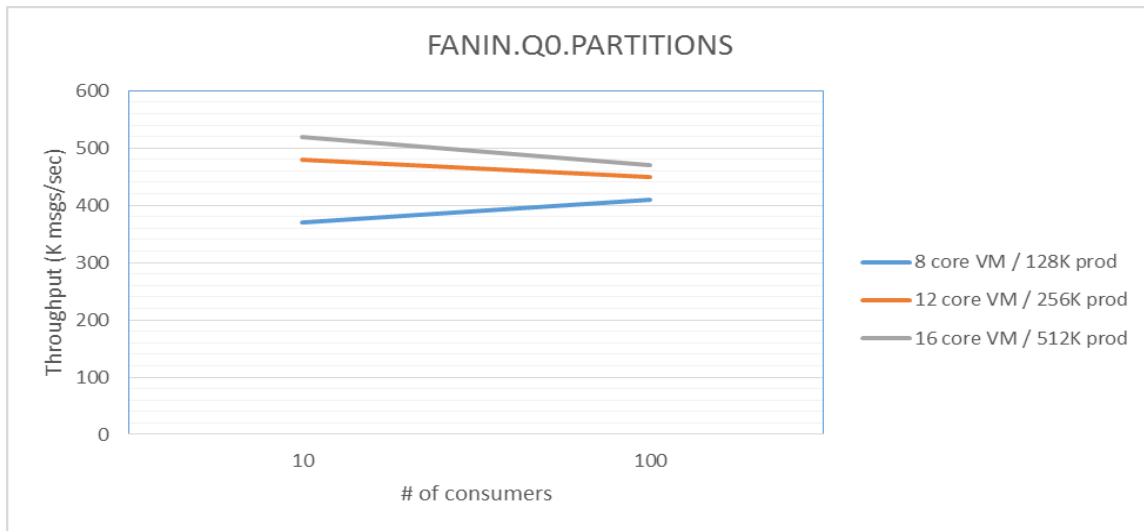
² The connection latency results for all CONNBURST tests are heavily affected by the number of consumer connections in the test. See the [Additional test results data](#) for the CONNBURST benchmarks.

Additional test results data

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

FANIN.Q0.PARTITIONS

The chart below compares the throughput results from the FANIN.Q0.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. In all three Virtual Edition configurations (8 core with 128K producers, 12 core with 256K producers, and 16 core with 512K producers) there was a slight improvement in throughput when distributing the load across 100 consumers instead of only 10 except for 16 core with 512K producers which had a 10% degradation.

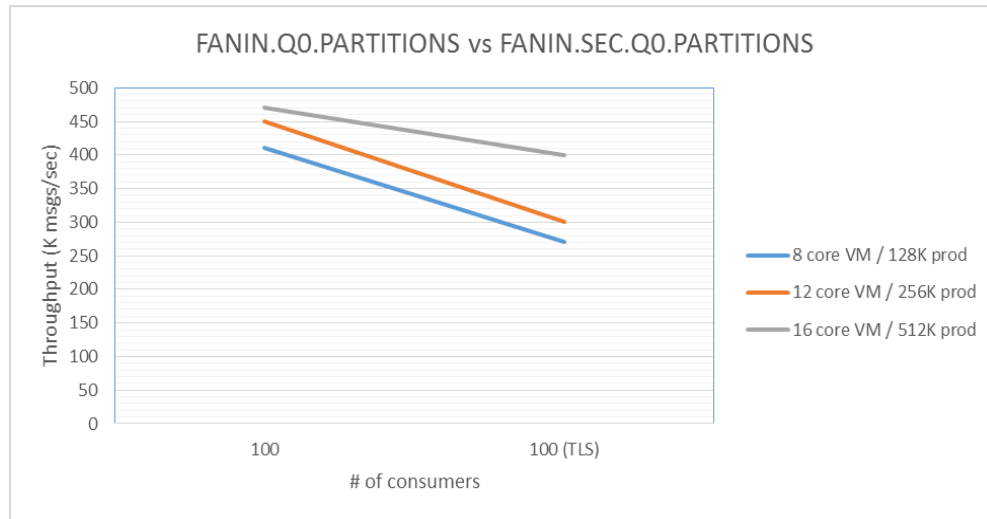


The table below lists the high water mark (HWM) stats from the FANIN.Q0.PARTITIONS test. These results show that when dividing the workload across a smaller number of subscriptions (e.g. 10 subscriptions) the subscription queues must be configured with larger limits, that is, in the messaging policies governing these subscriptions queues, the MaxMessages configuration parameter should be larger than a similar workload with 100 subscriptions.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	1288	7825	8424
100	144	275	223

FANIN.SEC.Q0.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q0.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions when the producers use a secured TLS communication with MessageSight. These results clearly show that when the producer connections are secured with TLS communications there is a performance degradation for all 3 configurations.

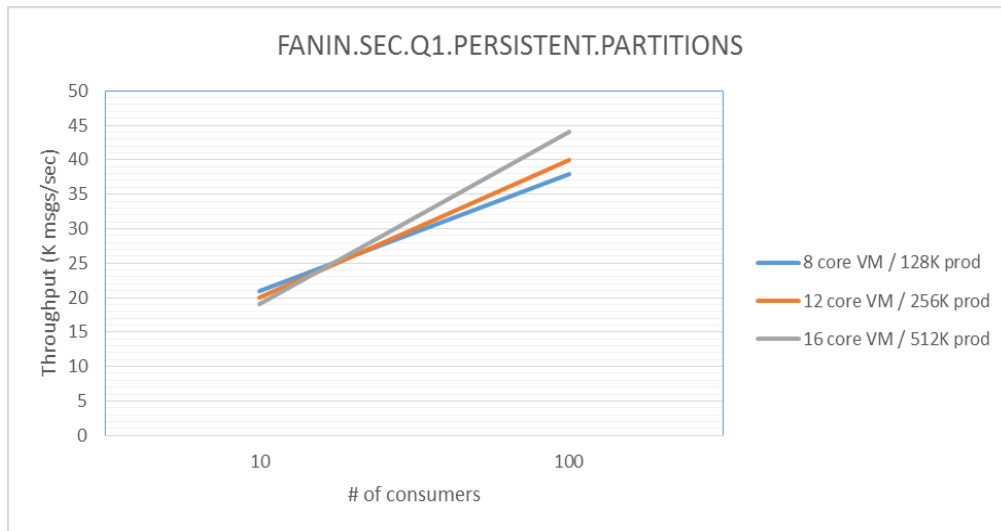


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q0.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
100	77	66	125

FANIN.SEC.Q1.PERSISTENT.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q1.PERSISTENT.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results clearly show that increasing the number of partitions/consumers in the workload that there is an improvement (100%) in the throughput.

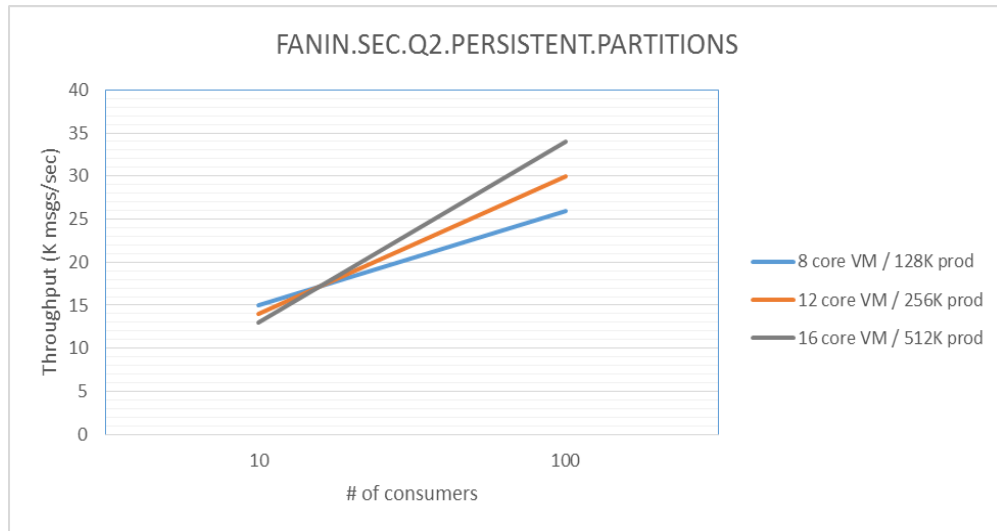


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q1.PERSISTENT.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	347	706	970
100	111	280	462

FANIN.SEC.Q2.PERSISTENT.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q2.PERSISTENT.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results clearly show a slight improvement when increasing the number of partitions/consumers in the workload. When comparing QoS level 1 to QoS level 2 messaging with partitions is evident that QoS 2 causes a degradation in throughput.

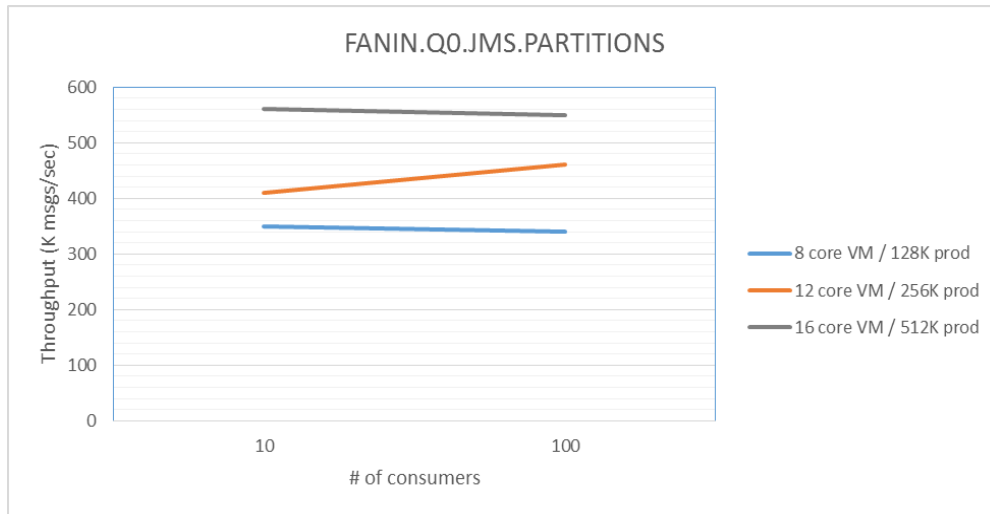


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q2.PERSISTENT.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	429	712	1099
100	83	317	909

FANIN.Q0.JMS.PARTITIONS

The chart below compares the throughput results from the FANIN.Q0.JMS.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results show that there is a slight degradation (5%) with 100 consumers versus 10 consumers for the 8 core with 128K producers and 16 core with 512K producers. Whereas the 12 core with 256K producers had a 10% improvement in throughput.

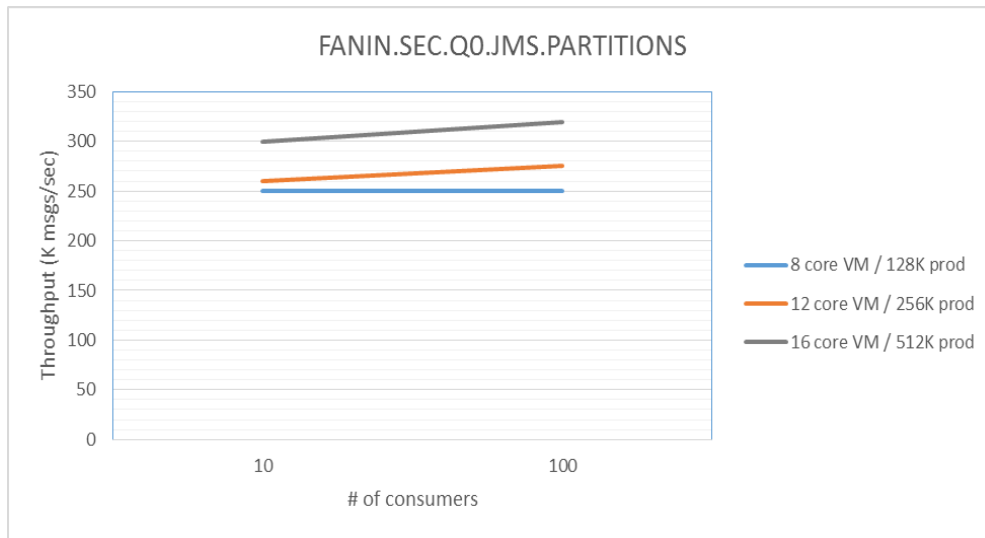


The table below lists the high water mark (HWM) stats from the FANIN.Q0.JMS.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	4374	3812	27686
100	216	6075	11484

FANIN.SEC.Q0.JMS.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q0.JMS.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results clearly show that when the producer connections are secured with TLS communications there is little or no performance benefit that can be observed by increasing the number of partitions/consumers in the workload.

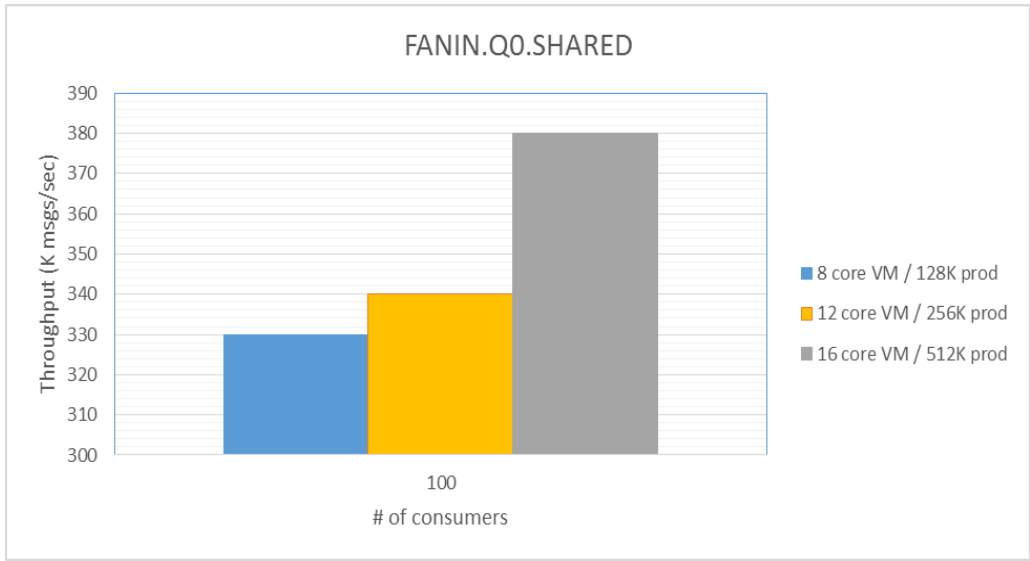


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q0.JMS.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	785	1229	13871
100	122	145	315

FANIN.Q0.SHARED

The chart below compares the throughput results from the FANIN.Q0.SHARED test. These results clearly show increasing the number of cores and memory increases the throughput rate.

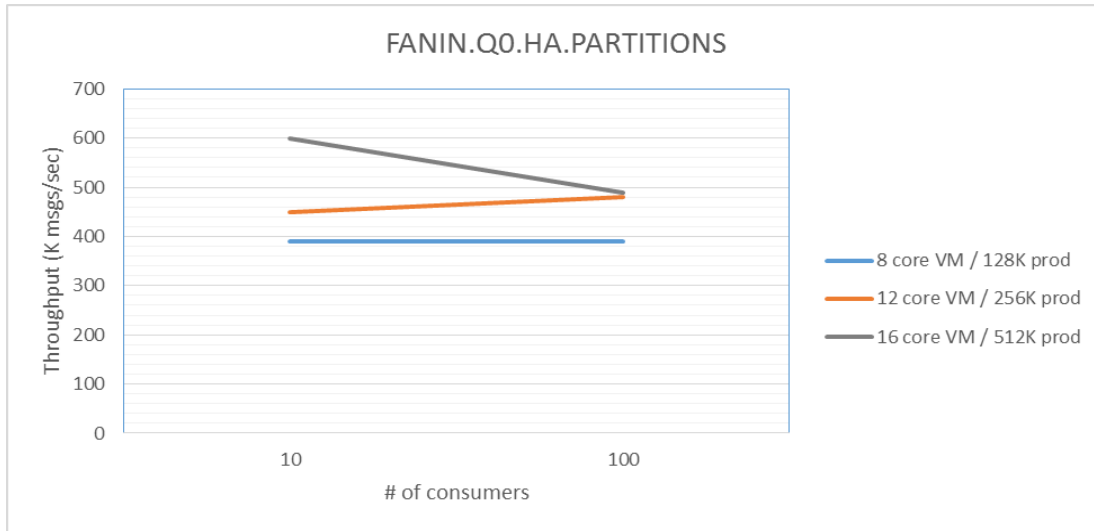


The table below lists the high water mark (HWM) stats from the FANIN.Q0.SHARED test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
100	1502	5335	3871

FANIN.Q0.HA.PARTITIONS

The chart below compares the throughput results from the FANIN.Q0.HA.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results clearly show that HA doesn't impact the throughput (see FANIN.Q0.PARTITIONS).

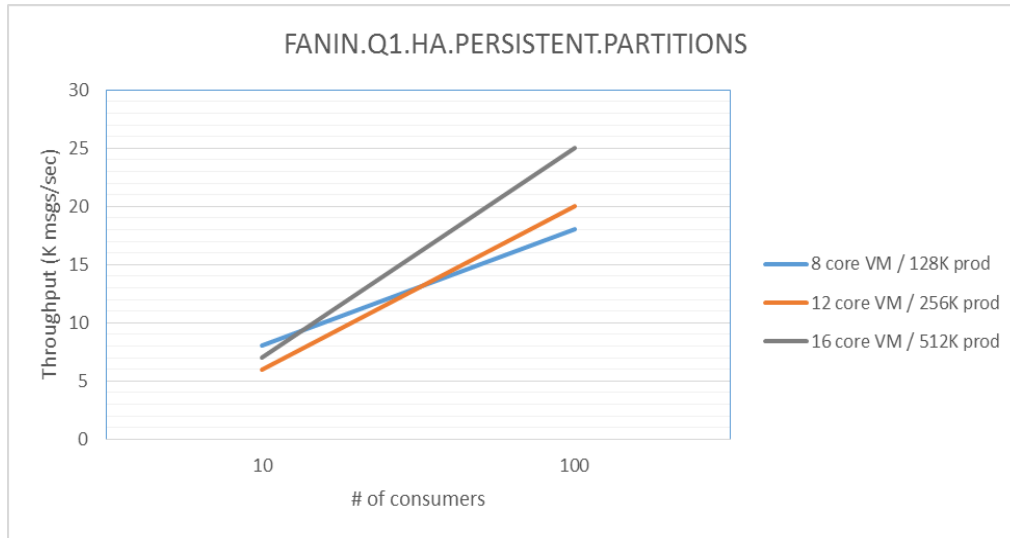


The table below lists the high water mark (HWM) stats from the FANIN.Q0.HA.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	1720	2826	2257
100	128	312	407

FANIN.Q1.HA.PERSISTENT.PARTITIONS

The chart below compares the throughput results from the FANIN.Q1.HA.PERSISTENT.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions. These results clearly show that there is a performance benefit that can be observed by increasing the number of partitions/consumers in the workload.

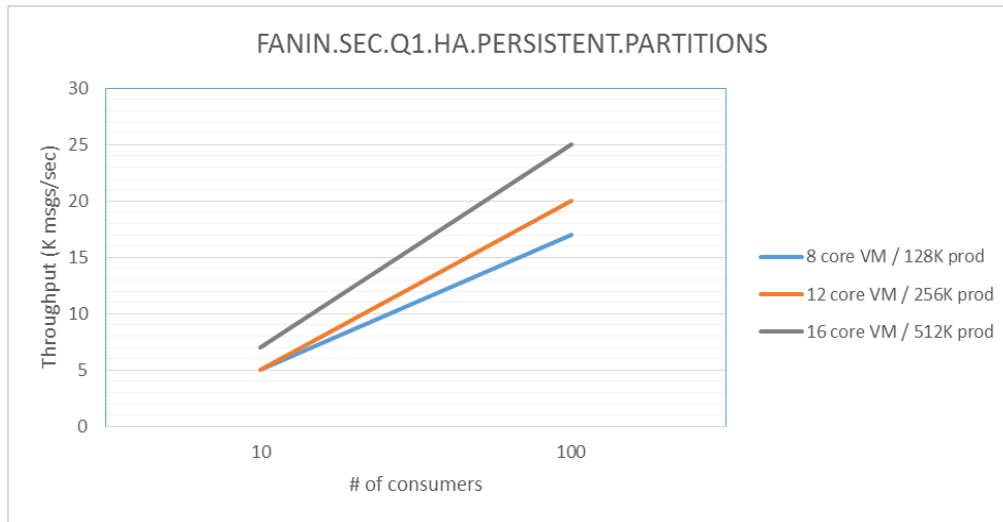


The table below lists the high water mark (HWM) stats from the FANIN.Q1.HA.PERSISTENT.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	275	221	459
100	169	172	246

FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single consumer, versus dividing the topic tree into 100 partitions using secured connections between the producers and MessageSight using a High Availability Pair (HA-Pair). These results when compared to the non-secured test (FANIN.Q1.HA.PERSISTENT.PARTITIONS) clearly show that when the producer connections are secured with TLS communications there is no performance benefit. The performance benefit observed is due to the increasing the number of partitions/consumers in the workload.

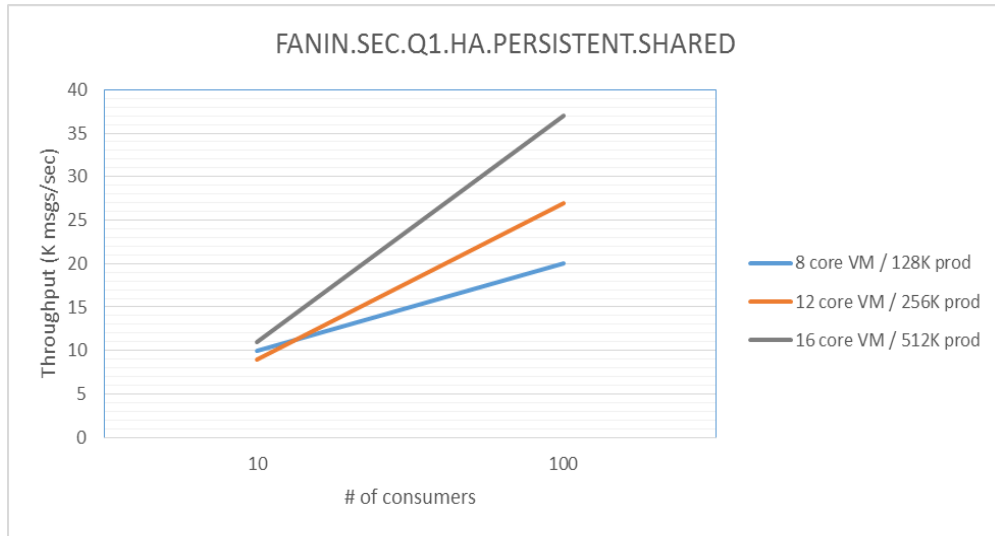


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	89	149	535
100	138	212	299

FANIN.SEC.Q1.HA.PERSISTENT.SHARED

The chart below compares the throughput results from the FANIN.SEC.Q1.HA.PERSISTENT.SHARED test when using a MQTT Shared Subscription with 10 consumers versus 100 consumers while using a High Availability Pair. These results clearly show that there is a performance improvement by increasing the number of partitions/consumers in the workload.

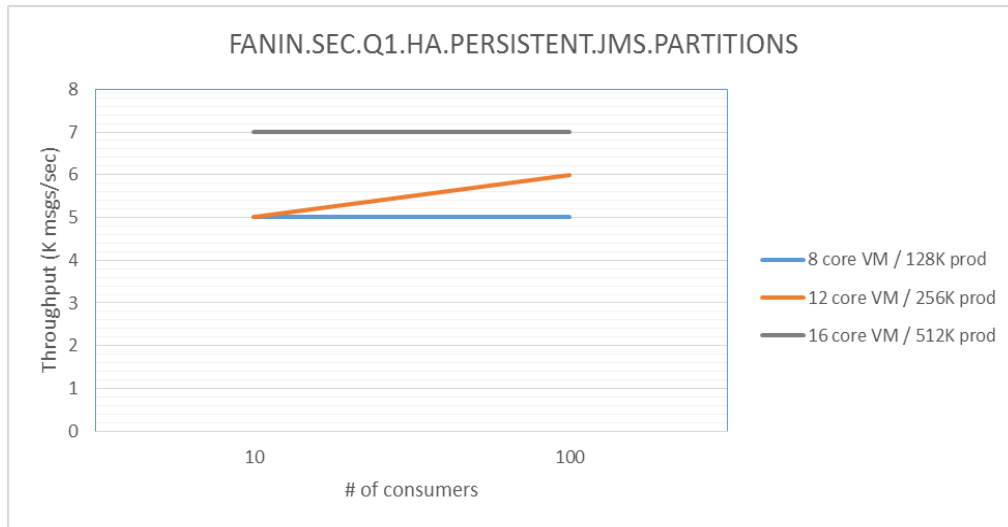


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q1.HA.PERSISTENT.SHARED test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	6835	2785	6232
100	13718	23330	30683

FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS

The chart below compares the throughput results from the FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS test when dividing the topic tree into 10 partitions, each partition subscribed to by a single JMS consumer, versus dividing the topic tree into 100 partitions while using an High Availability Pair. These results clearly show that there is little or no performance benefit that can be observed by increasing the number of partitions/consumers in the workload.

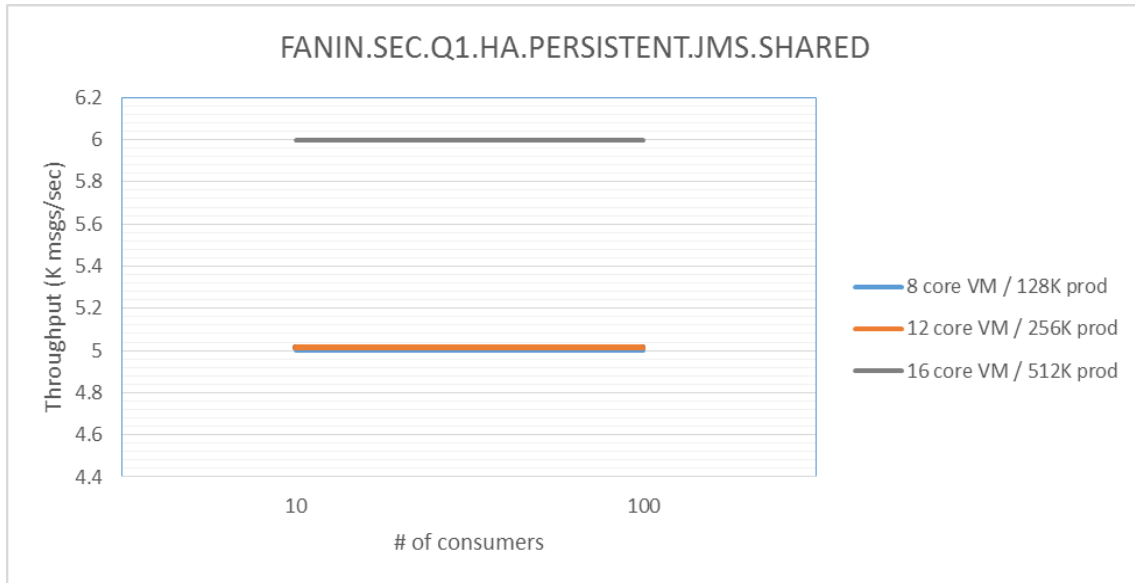


The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	436	321	1007
100	184	228	218

FANIN.SEC.Q1.HA.PERSISTENT.JMS.SHARED

The chart below compares the throughput results from the FANIN.SEC.Q1.PERSISTENT.JMS.SHARED test when using a JMS Shared Subscription by 10 consumers versus 100 consumers while using High Availability Pair. These results clearly show that there is no performance benefit that can be observed whether using 10 or 100 consumers.



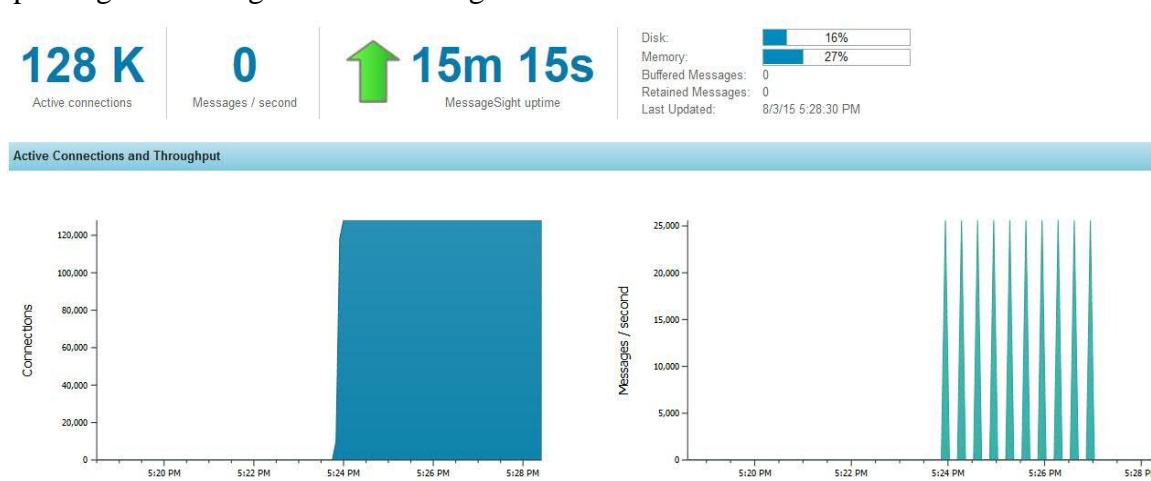
The table below lists the high water mark (HWM) stats from the FANIN.SEC.Q1.HA.PERSISTENT.JMS.SHARED test.

# of consumers	8 core VM HWM (msgs)	12 core VM HWM (msgs)	16 core VM HWM (msgs)
10	1686	2853	1020
100	9247	9369	9042

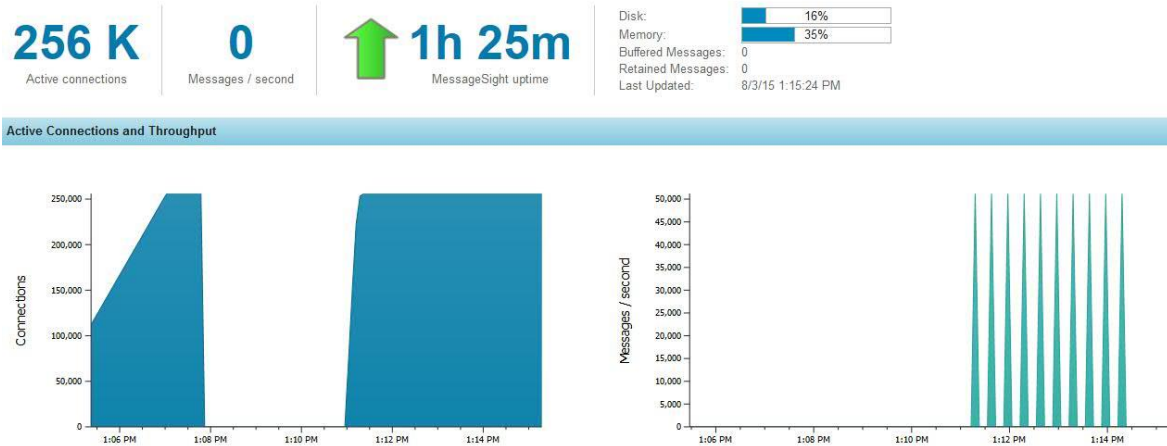
BROADCAST

The series of screenshots below are from the Appliance Dashboard on the MessageSight WebUI. The throughput graph on the right displays the outgoing (egress) message rate during the BROADCAST benchmarks. Each spike is a single message, published from the lone producer, and being delivered to all subscribers of the *broadcast* topic.

The figure below shows an 8 core VM with 128K consumers.
The peak egress message rate = 25K msgs/sec



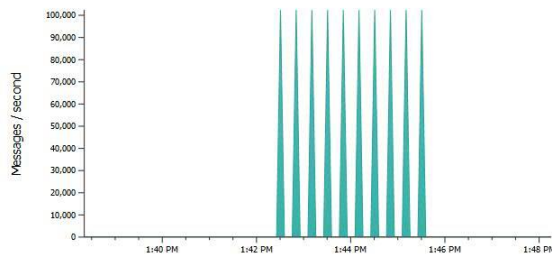
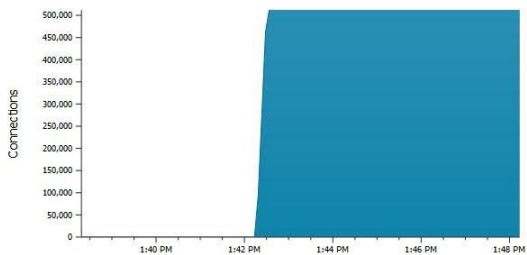
The figure below shows a 12 core VM with 256K consumers.
The peak egress message rate = 50K msgs/sec



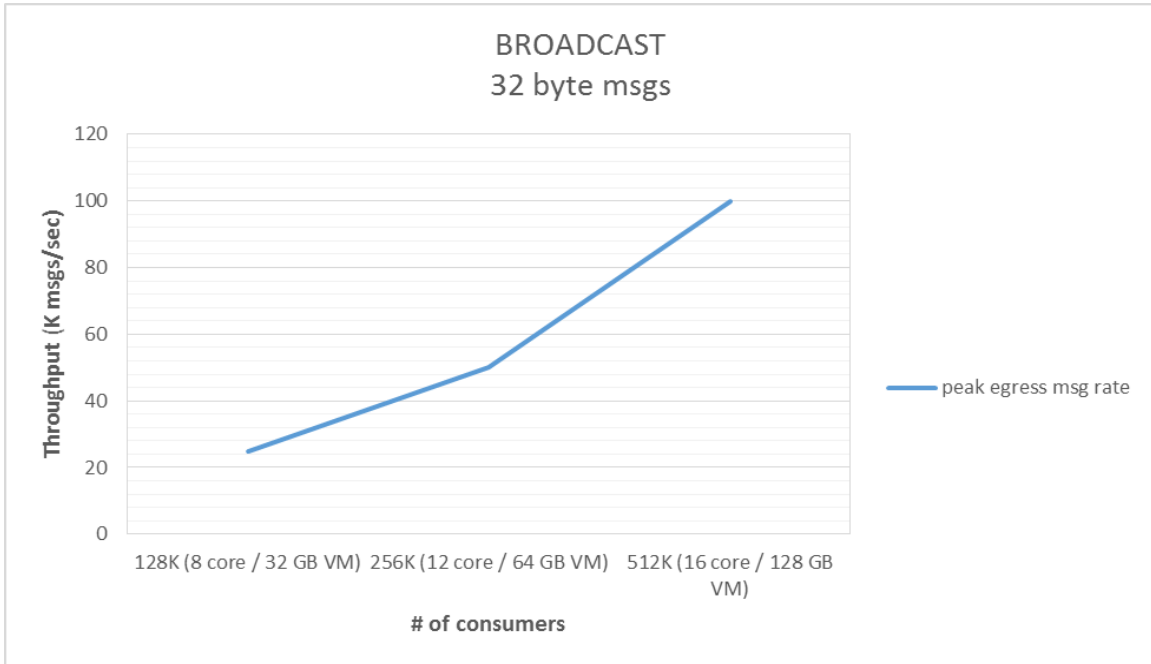
The figure below shows a 16 core VM with 512K consumers.
The peak egress message rate = 100K msgs/sec

512 K Active connections	0 Messages / second	 21m 56s MessageSight uptime	Disk: 10% Memory: 25% Buffered Messages: 0 Retained Messages: 0 Last Updated: 8/3/15 1:48:15 PM
------------------------------------	-------------------------------	---	---

Active Connections and Throughput



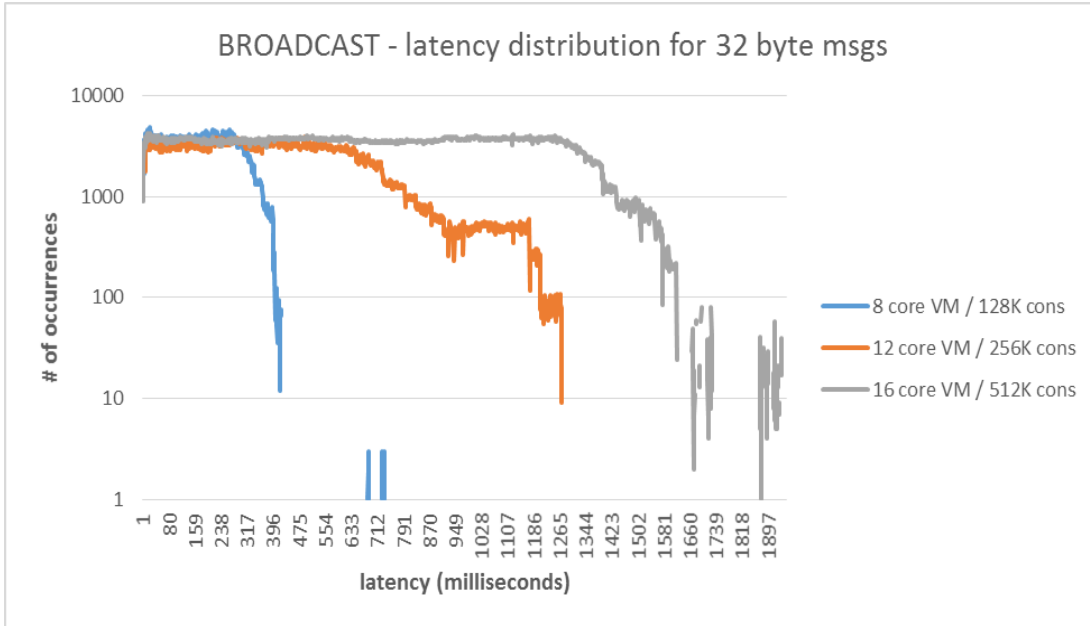
The chart below summarizes the peak egress message rate results of the broadcast benchmark for 32 byte messages.



The table below summarizes the BROADCAST benchmark peak egress message rate results across three different message sizes (32 bytes, 128 bytes, and 512 bytes). The results below show that there is no difference in peak egress message rate across the message sizes tested.

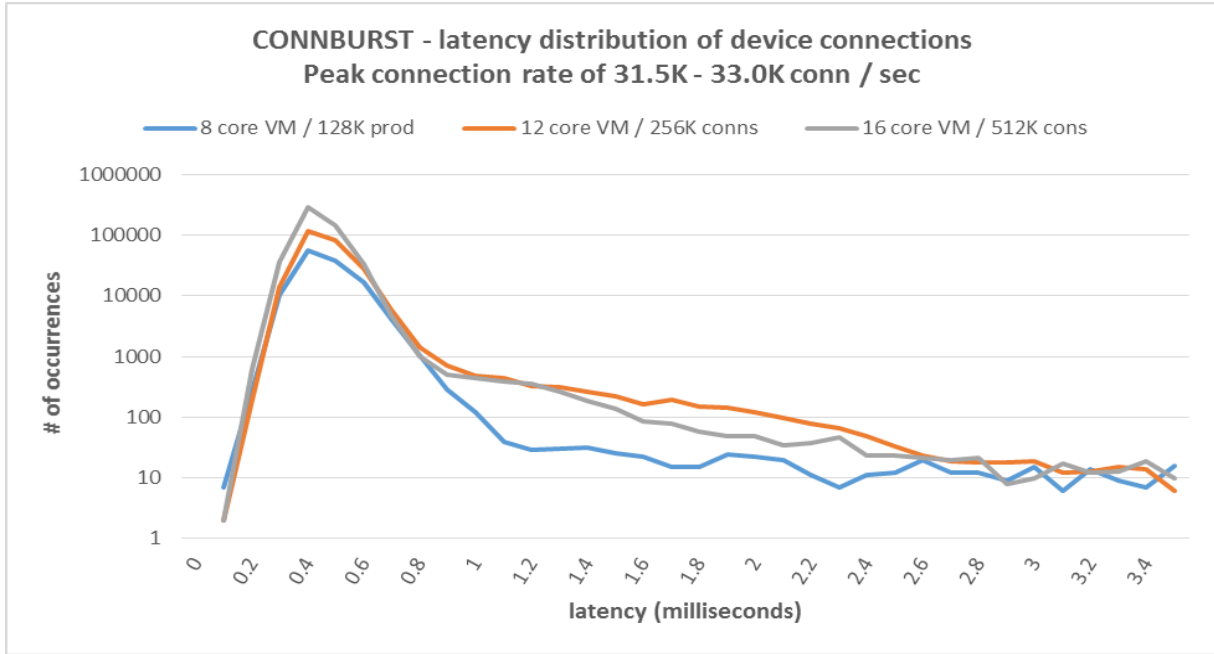
Msg Size (bytes)	8 core VM / 128K consumers (msgs/sec)	12 core VM / 256K consumers (msgs/sec)	16 core VM / 512K consumers (msgs/sec)
32	25K	50K	100K
128	25K	50K	100K
512	25K	50K	100K

The chart below compares the latency distribution of the message latency in the BROADCAST benchmark across the three MessageSight VM sizes. Message latency is defined as the time when the consumer received the “broadcast” message minus the time when the message was sent by the lone producer.

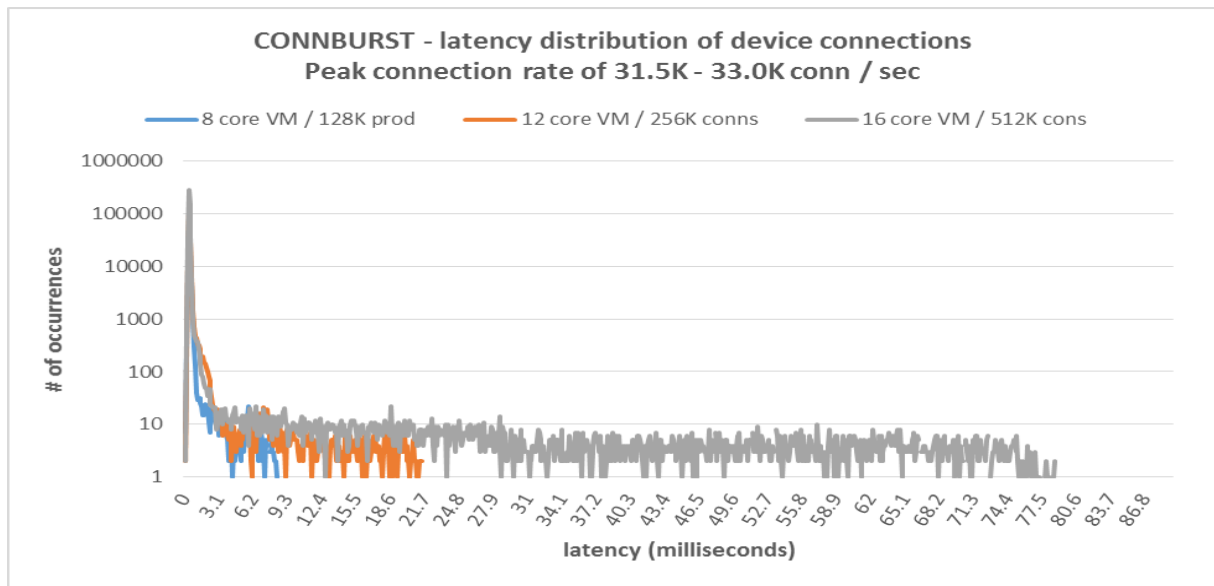


CONNBURST

The chart below compares the distribution of connection latency in the CONNBURST benchmark across the three MessageSight VM sizes. Connection latency is defined as the time when the consumer sends the TCP SYN request to the time when it receives an MQTT CONNACK from the MessageSight appliance.



The chart below shows the tail of the latency distribution for the CONNBURST test. The chart below also shows that the 512K connection tests has a broader latency distribution than 128K and 256K connections.

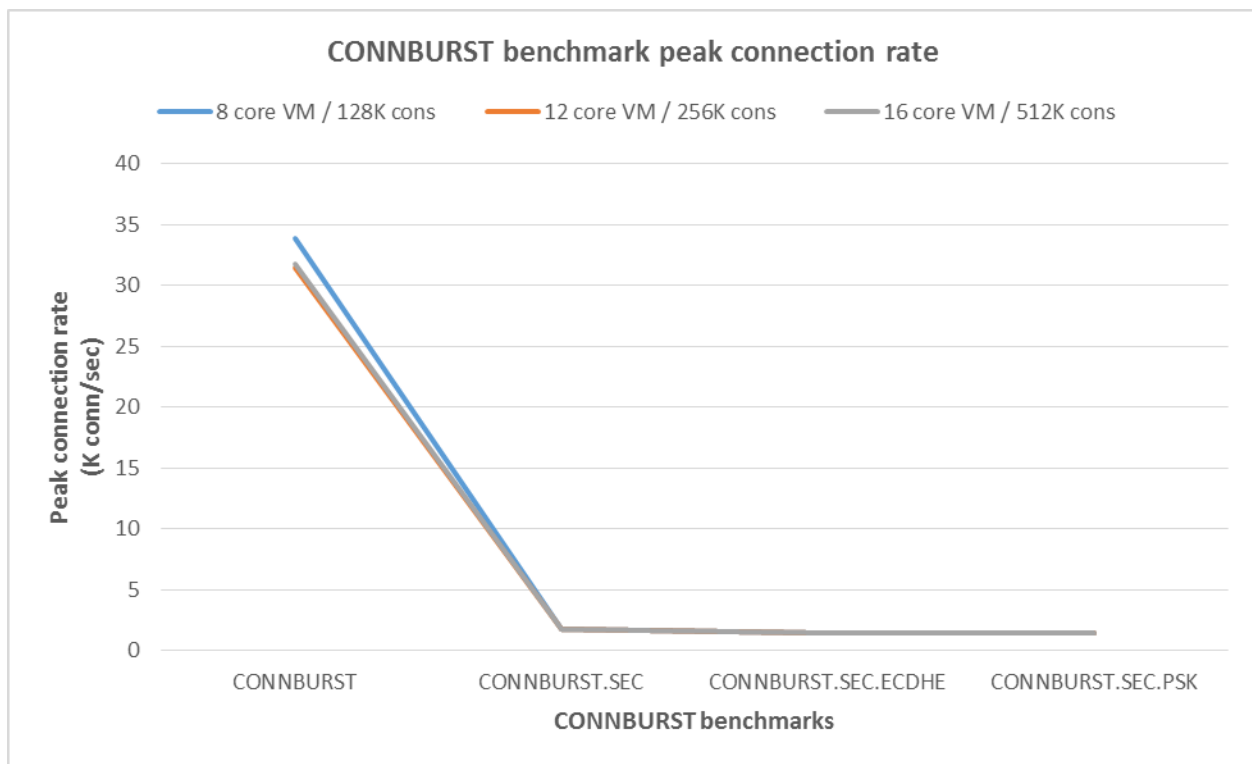


CONNBURST.SEC, CONNBURST.SEC.ECDHE, CONNBURST.SEC.PSK

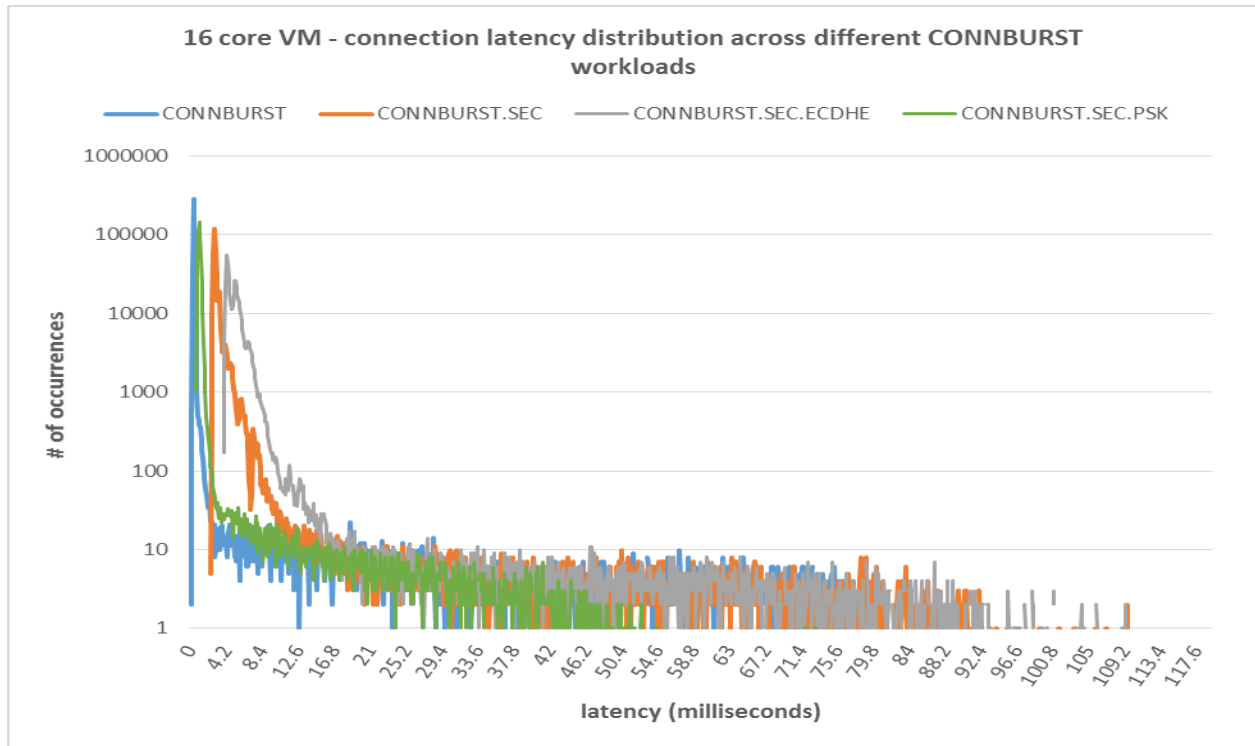
These three benchmarks measure the peak connection rate sustained by IBM MessageSight when the device connections are secured using TLS which terminate on the MessageSight server. All three benchmarks use the TLSv1.2 protocol. For the Ciphers and key sizes used in each test, see the table below.

Benchmark	Cipher	Key size (bits)
CONNBURST.SEC	AES256-GCM-SHA384	2K
CONNBURST.SEC.ECDHE	ECDHE-RSA-AES256-GCM-SHA384	2K
CONNBURST.SEC.PSK	PSK-AES256-CBC-SHA	128

The chart below compares the peak connection rate of all CONNBURST benchmarks.



The chart below compares the latency distribution of the different CONNBURST benchmarks on a 16 core MessageSight VM.



Appendix A: IBM MessageSight configuration

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

Network interface configuration

The Configuring Switches and mapping virtual network interfaces section above describes the network interface configuration used for the MessageSight Virtual Edition benchmarks. Both clients and MessageSight interfaces were configured with IPv4 addresses with a 23 bit subnet mask. Client systems required multiple IPv4 addresses per interface to accommodate the large number of connections in these workloads.

HA replication traffic was configured to use a separate interface from the client-MessageSight data traffic.

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.

Deeper subscription queues are required for the MessageSight Virtual Edition in order to compensate for intermittent scheduling delays in the hypervisor. Subscription queue sizes are configured on the messaging policy object associated with the subscription by setting the MaxMessages configuration parameter.

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).

Certificate profile configuration

A single certificate profile was created. The server certificate and key used in all TLS *SEC* test cases, except CONNBURST.SEC.PSK is 2048 bits. In the CONNBURST.SEC.PSK the key size used was 128 bit.

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	1000	
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.

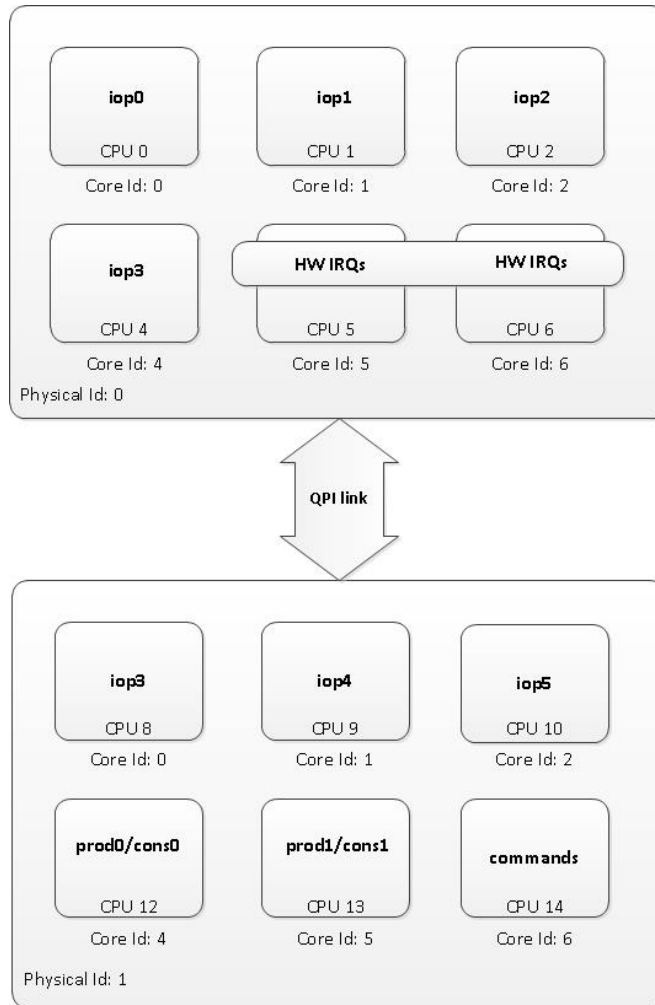
Services stopped: iptables, ip6tables, cpuspeed, cups, crond, atd, auditd, autofs, rhnsd, rhsmcrtcd, postfix, mdmonitor, jexec, netfs, and more

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 CPU affinity

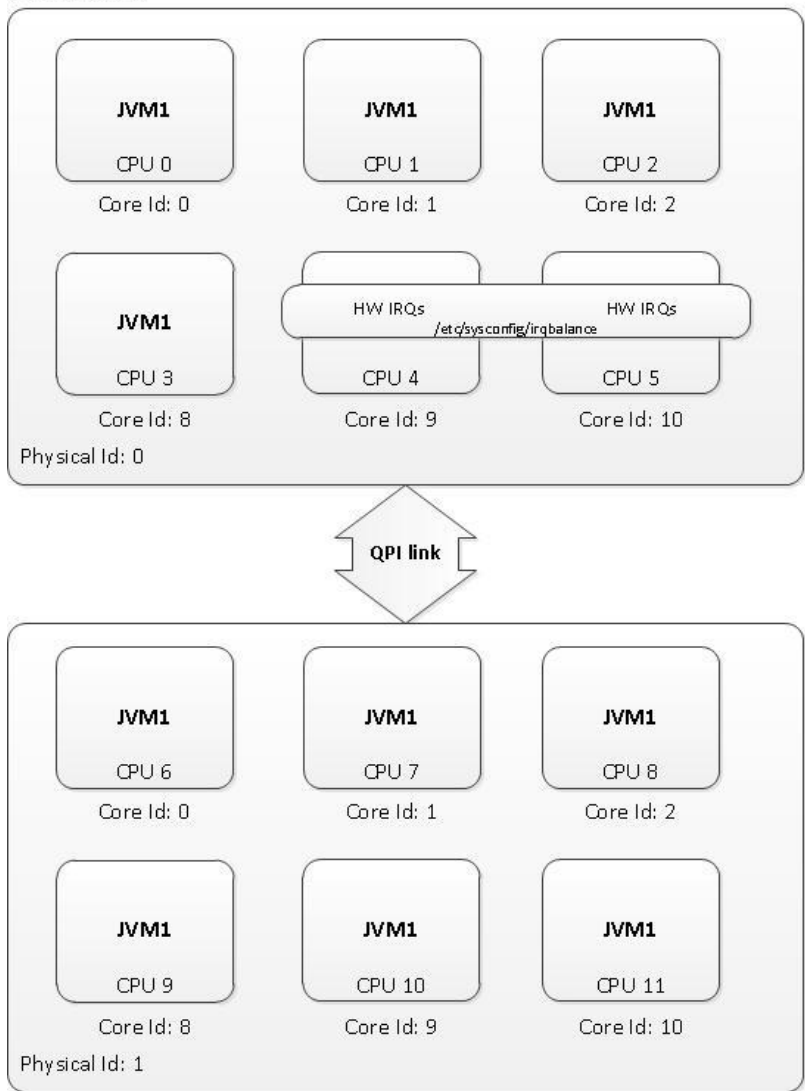
The following diagram shows how the *mqttbench* test harness threads were pinned to the CPUs of the client machine. The diagram below represents the CPU to core mapping on the client machines used in the benchmark tests. *mqttbench* was configured with 6 IOP threads and 1 IOL thread in these benchmarks.



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.



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: 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.Q0.SINGLE

Machine 1 (consumer)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort=16901
export IMAServer=10.10.0.222
```

Invocation:

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

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/t/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

$X * 2$ = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.SEC.Q0.JMS.SINGLE

Machine 1 (consumer)

```
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
export IMAServer=10.10.0.222
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:0:3:2:0:/p/#:1:1:1:1 -mt t
```

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -r Y -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.Q0.PARTITIONS /
FANIN.Q0.HA.PARTITIONS**

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:/p/#:1:X:2 -np Y
```

Where X is 5 or 50 and Y is 10 or 100. This benchmark was run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100, which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.Q[1|2].PERSISTENT.PARTITIONS /
FANIN.Q1.HA.PERSISTENT.PARTITIONS**

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation for MQTT QoS 1 & 2:

```
QoS 1: ./imacInt.sh -d 0 -rx 1:0:/p/#:1:X:2 -np Y
QoS 2: ./imacInt.sh -d 0 -rx 2:0:/p/#:1:X:2 -np Y
```

Where X is 5 or 50 and Y is 10 or 100. This benchmark is run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 & 2:

```
QoS 1: ./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
QoS 2: ./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100, which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.SEC.Q0.PARTITIONS

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:/p/#:1:X:2 -np Y
```

Where X is 5 or 50 and Y is 10 or 100. This benchmark was ran with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100, which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.SEC.Q[1|2].PERSISTENT.PARTITIONS /
FANIN.SEC.Q1.HA.PERSISTENT.PARTITIONS**

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation for MQTT QoS 1 & 2:

```
QoS 1: ./imacInt.sh -d 0 -rx 1:0:/p/#:1:X:2 -np Y
QoS 2: ./imacInt.sh -d 0 -rx 2:0:/p/#:1:X:2 -np Y
```

Where X is 5 or 50 and Y is 10 or 100. This benchmark is run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 & 2:

```
QoS 1: ./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
QoS 2: ./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100, which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.Q0.JMS.PARTITIONS

Machine 1 (consumers)

```
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"
export IMAServer="10.10.0.222 10.12.0.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:0:3:2:0:/p/#:1:1:X:10 -mt t -np Y
```

Where X is 1 or 10 and Y is 10 or 100. This benchmark is run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901 16902 16901
                16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100 which is determined on whether there is 10 or 100 consumer connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.SEC.Q0.JMS.PARTITIONS**Machine 1 (consumers)**

```
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"
export IMAServer="10.10.0.222 10.12.0.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:0:3:2:0:/p/#:1:1:X:10 -mt t -np Y
```

Where X is 1 or 10 and Y is 10 or 100. This benchmark is run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100 which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.SEC.Q[1|2].PERSISTENT.JMS.PARTITIONS /
FANIN.SEC.Q1.HA.PERSISTENT.JMS.PARTITIONS**

Machine 1 (consumers)

```
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"
export IMAServer="10.10.0.222 10.10.2.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:100:3:2:1:/p/#:1:1:X:10 -mt t -np Y
```

Where X is 1 or 10 and Y is 10 or 100. This benchmark was run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 & 2:

```
QoS 1: ./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
QoS 2: ./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -np Y -r Z -s 32-32 -wr -rrs
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is 10 or 100 which is determined on whether there is 10 or 100 consumers connections/topic tree partitions.

Z is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.Q0.SHARED

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:\$SharedSubscription/a/#:1:50:2
```

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.Q[1|2].PERSISTENT.SHARED

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:\$SharedSubscription/a/#:1:50:2
```

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 and 2:

QoS 1:

```
./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

QoS 2:

```
./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.SEC.Q0.SHARED

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:\$SharedSubscription/a/#:1:50:2
```

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.SEC.Q[1|2].PERSISTENT.SHARED /
FANIN.SEC.Q1.HA.PERSISTENT.SHARED**

Machine 1 (consumers)

```
export IMAClient=MQTT
export LargeConn=0
export BatchingDelay=1
export IMAPort="16901 16902"
export IMAServer="10.10.1.22:10.10.0.222 10.10.3.22:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:\$SharedSubscription/a/#:1:50:2
```

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 and 2:

QoS 1:

```
./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

QoS 2:

```
./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.SEC.Q0.JMS.SHARED

Machine 1 (consumers)

```
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"
export IMAServer="10.10.0.222 10.12.0.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:0:3:2:0:/p/#:1:1:X:10 -mt ss
```

Where X is 1 or 10. This benchmark was run with 10 and 100 consumer connections/topic tree partitions.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

FANIN.Q[1|2].PERSISTENT.JMS.SHARED

Machine 1 (consumers)

```
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"
export IMAServer="10.10.0.222 10.12.0.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:100:3:2:1:/p/#:1:1:X:10 -mt ss
```

Where X is 1 or 10. This benchmark is run with 10 & 100 consumer connections.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 and 2:

QoS 1:

```
./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

QoS 2:

```
./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msgs/sec

**FANIN.SEC.Q[1|2].PERSISTENT.JMS.SHARED /
FANIN.SEC.Q1.HA.PERSISTENT.JMS.SHARED**

Machine 1 (consumers)

```
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"
export IMAServer="10.10.0.222 10.12.0.222"
```

Invocation:

```
taskset -c 0-3,6-11 ./imacInt.sh -d 0 -rx -1:100:3:2:1:/p/#:1:1:X:10 -mt ss
```

Where X is 1 or 10. This benchmark is run with 10 & 100 consumer connections.

Machine 2 (producers)

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=500
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation for MQTT QoS 1 and 2:

QoS 1:

```
./imacInt.sh -d 0 -tx 1:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

QoS 2:

```
./imacInt.sh -d 0 -tx 2:0:/p/t:1:X:2 -r Y -s 32-32 -wr
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of producer connections created in this test.

Y is the aggregate message rate sent by the producers and is expressed in units of msg/sec

BROADCAST

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
              16902 16901 16902 16901 16902 16901 16902"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                 10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                 10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                 10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                 10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                 10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                 10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                 10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -tx 0:0:t:1:1 -rx 0:1:t:1:X:2 -wr -rrs -s 32-32 -c 10 -r 0.05
            -T 0x81 -u 1.0e-3 -lcsv broadcast_32_qos0.csv
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

X * 2 = total number of consumer connections created in this test.

-u 1e-3 = latency measurements are stored in units of milliseconds

CONNBURST

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=5
export BatchingDelay=1
export UseSecureConn=0

export IMAPort="16901 16902 16901 16902 16901 16902 16901 16902 16901
               16902 16901 16902 16901 16902 16901 16902 16901"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                 10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                 10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                 10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                 10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                 10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                 10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                 10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:t:1:X:2 -T 0xA0 -u 1.0e-4 -lcsv connburst.csv
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

-u 1e-4 = latency measurements are stored in units of tens of milliseconds

CONNBURST.SEC

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1000
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
               16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                 10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                 10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                 10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                 10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                 10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                 10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                 10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:t:1:X:2 -T 0xA0 -u 1.0e-4 -lcsv connburst_sec.csv
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

-u 1e-4 = latency measurements are stored in units of tens of milliseconds

CONNBURST.SEC.ECDHE

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1300
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=ECDHE-RSA-AES256-GCM-SHA384
export SSLClientMeth=TLSv12
export CERTSIZE=2048

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                  10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                  10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                  10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                  10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                  10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                  10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                  10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:t:1:X:2 -T 0xA0 -u 1.0e-4 -lcsv connburst_sec_ecdhe.csv
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

-u 1e-4 = latency measurements are stored in units of tens of milliseconds

CONNBURST.SEC.PSK

Machine 1

```
export IMAClient=MQTT
export LargeConn=1
export DelayCount=1
export DelayTime=1300
export BatchingDelay=1
export UseSecureConn=1
export SSLCipher=PSK-AES256-CBC-SHA
export SSLClientMeth=TLSv12

export IMAPort="16903 16904 16903 16904 16903 16904 16903 16904 16903
                16904 16903 16904 16903 16904 16903 16904"

export IMAServer="10.10.1.81:10.10.0.222 10.10.3.81:10.10.0.222
                 10.10.5.81:10.10.0.222 10.10.7.81:10.10.0.222
                 10.10.9.81:10.10.0.222 10.10.11.81:10.10.0.222
                 10.10.13.81:10.10.0.222 10.10.15.81:10.10.0.222
                 10.10.17.81:10.10.0.222 10.10.19.81:10.10.0.222
                 10.10.21.81:10.10.0.222 10.10.23.81:10.10.0.222
                 10.10.25.81:10.10.0.222 10.10.27.81:10.10.0.222
                 10.10.29.81:10.10.0.222 10.10.31.81:10.10.0.222"
```

Invocation:

```
./imacInt.sh -d 0 -rx 0:1:t:1:X:2 -T 0xA0 -u 1.0e-4 -lcsv connburst_sec_psk.csv
            -psk psk.csv
```

Where X is derived from the amount of memory allocated to the MessageSight VM.

-u 1e-4 = latency measurements are stored in units of tens of milliseconds

The PreSharedKey file was uploaded to MessageSight and applied with the following command: `imaserver apply PreSharedKey "PSKFile=psk.csv"`