Performance testing on AWS featuring SIQM, NativeHA and NHA:CRR scenarios

Objective

In previous OCP reports, data has been published from scenarios running in a bare-metal OpenShift Cluster in our Hursley datacenter. Many customers are looking to run their messaging workload both on-premise and in cloud environments. This paper will look at the performance of running MQ Queue Managers (QM) within OpenShift in AWS.

There are three main ways of running OpenShift in AWS:

- RedHat OpenShift Service on AWS (ROSA)
- RedHat OpenShift dedicated service
- OpenShift installed on EC2 instances

This Whitepaper will look at option 3 using Installer Provisioned Infrastructure (IPI), as it allows full control of compute and storage options and the report will focus on popular choices in production use.

The NHA:CRR testing will consist of an OpenShift cluster in both US-east-2(N. Virginia) and US-west-2(N. California) locations and demonstrate the performance seen in a highly available distributed environment.

To assist in sizing and to show the impact of increasing availability, results will be shown for Single Instance Queue Manager (SIQM), NativeHA (NHA) and NativeHA:Cross Region Replication (NHA:CRR).

The test scenario is the Request/Responder scenario (RR-CC) as featured in our xLinux MQ Performance report. Each round trip is comprised of two message Puts and two message Gets and utilizes pairs of queues to communicate between the requester clients and responder applications.

Environment

OpenShift: 4.18.17
MQ Operator: 3.6
MQ: 9.4.3.0-r1
Control Plane nodes: 3
Worker nodes: 4 (3 for NHA, 1 for client application)
Instance specification: m5.4xlarge – 16vCPU, 64GB RAM, 10GbE
Storage specification: EBS GP3
Test client: cphtestp (containerized MQI test application, ran in different AZ to active QM)
Test client config: 12 cpu request, 16 cpu limit, 16GB memory request/limit, pinned to us-east-2b

Network
The latency between the US-east-2 and US-west-2 locations for the NHA:CRR testing is approximately 47ms. Latency measured between availability zones <1ms.

A single threaded iperf3 test between nodes in different Availability Zones(AZ) measured ~5Gbps. Additional threads could produce 10Gbps bandwidth as expected by the EC2 instance specification. A single thread test between US-east-2 and US-west-2 was measured at ~500Mbps

Storage

EBS GP3 storage was selected as it provides a good balance between cost and performance. 8GB volumes created for SIQM. 50GB volumes for NHA. 100GB volumes for the NHA:CRR remote group. We saw no difference in performance between encrypted and non-encrypted volumes, so settled on the following Storage class configuration:

GP3 Throughput: 1000MB/s

GP3 IOPS: 4000

Encrypted: true

A single thread performing an MQ simulated workload (MQLDT testing) achieved 64MB/s with 128KB block size. Increasing the block size and/or concurrency could push this towards 500MB/s which is approaching the limit of this EC2 instances storage bandwidth.

A similar test against AWS EFS storage produced 16MB/s with a 128KB block size, peaking at 100MB/s with 6 threads.

EBS gp3 does perform replication within an availability zone, and whilst it could be argued that it is not strictly necessary given that NativeHA is already replicating between zones, it may provide faster Pod/Node recovery than if instance storage was utilized and all data would need to be resynchronized after a failure.

SIQM Test configuration and results

This section uses the following test configuration:

- Persistent
- 2K / 20K / 200K messages size
- JSON formatted payload
- TLS12 enabled for Application communication
- 8GB PV allocated to QM instance
- AWS Network (between AZ): 5-10Gb
- Core allocation to QM pod resources: 8 cores

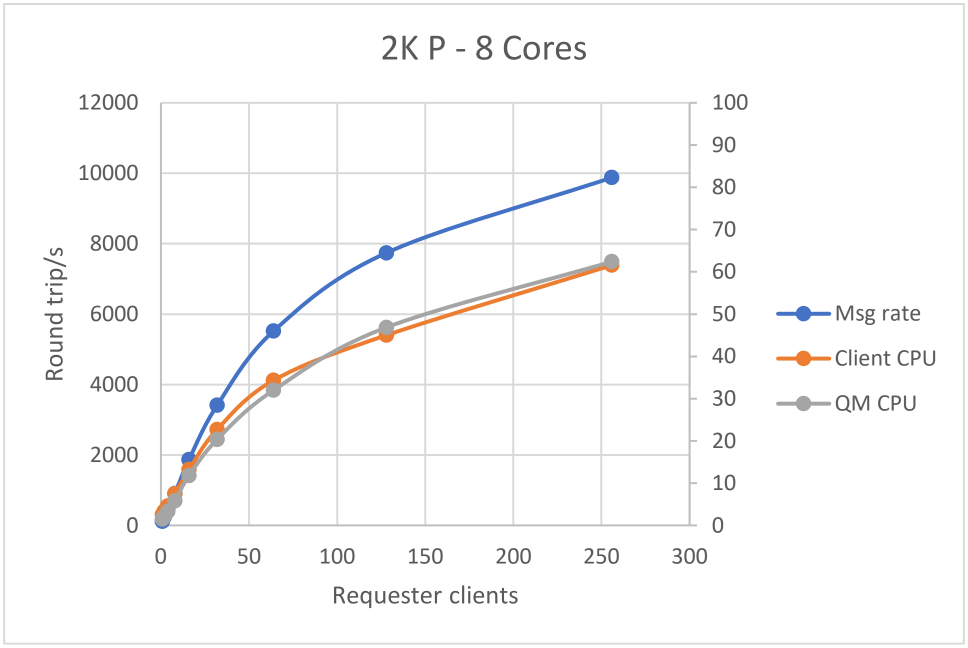


Figure 1 - 2K SIQM Persistent messaging

In the chart above for 2K Persistent messaging, the throughput of the QM approaches 10,000 round trip/s with 250 requester clients.

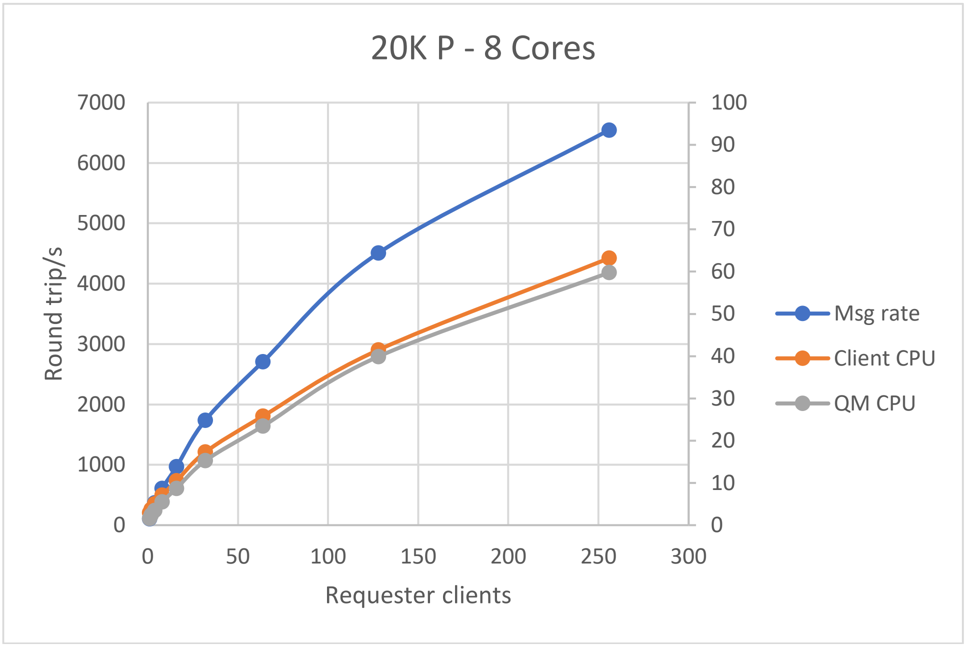


Figure 2 - 20K SIQM Persistent messaging

The chart above shows that as we increase the message size to 20K, the QM can process ~6,500 round trip/s at 250 requester clients.

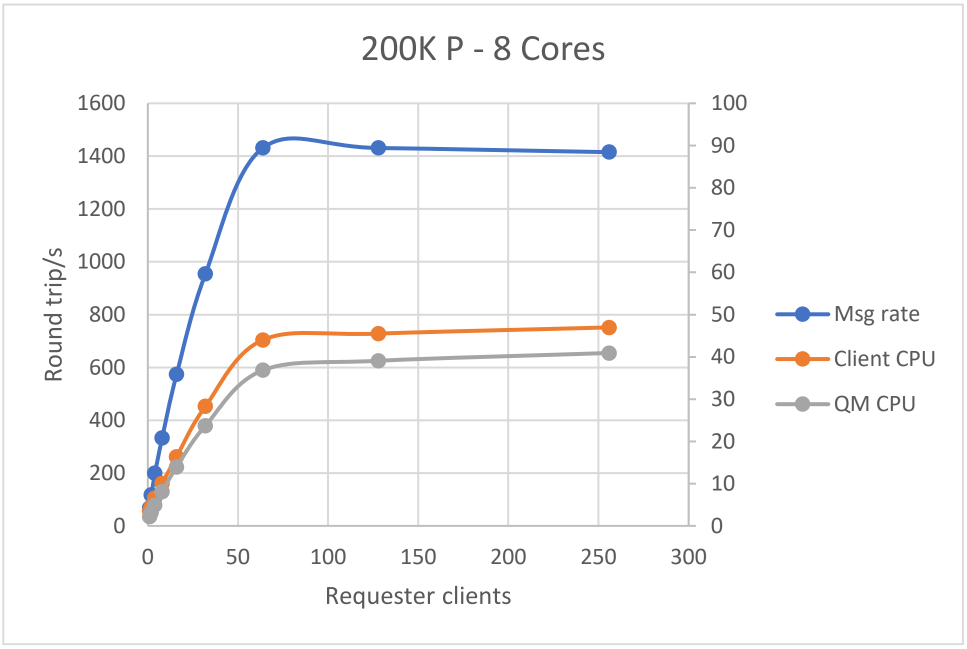


Figure 3 - 200K SIQM Persistent messaging

The above chart shows that with a 200KB message size, 64 threads can achieve ~1,400 round trip/s which is driving over 500MB/s through the EBS storage layer and reaching saturation point.

The base SIQM data shows good overall performance in AWS, though utilizing increasing numbers of threads to achieve peak throughput. IO saturation was reached in the 200K test.

NHA Test configuration and results

This section uses the following test configuration:

- Persistent
- 2K / 20K / 200K messages size
- JSON formatted payload
- TLS12 enabled for Application and Live HA Group communication
- 50GB PV allocated to Live group instances
- IMGLOGLN set to 12500MB (25% of PV) – Controls when MQ media images are taken
- LZ4 Compression Enabled, CompressionThreshold=64 within NHA group
- AWS Network (between AZ): 5-10Gb
- Core allocation to QM pod resources: 8 cores

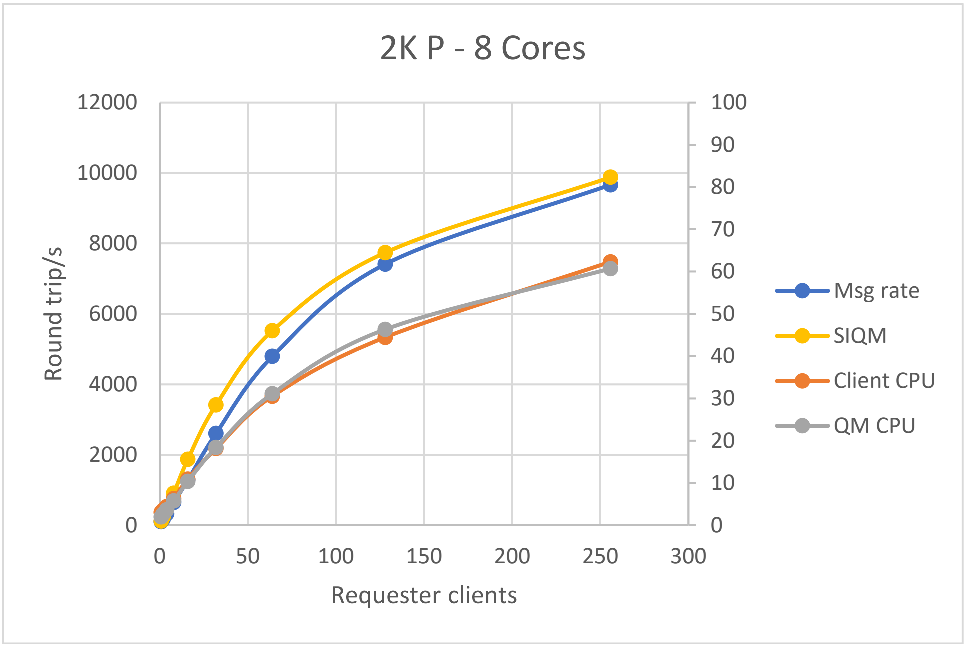


Figure 4 - 2K NHA Persistent messaging

The above chart shows how closely NHA performs in comparison to the SIQM performance with 2K message size, with data now being persisted in 3 different AZ.

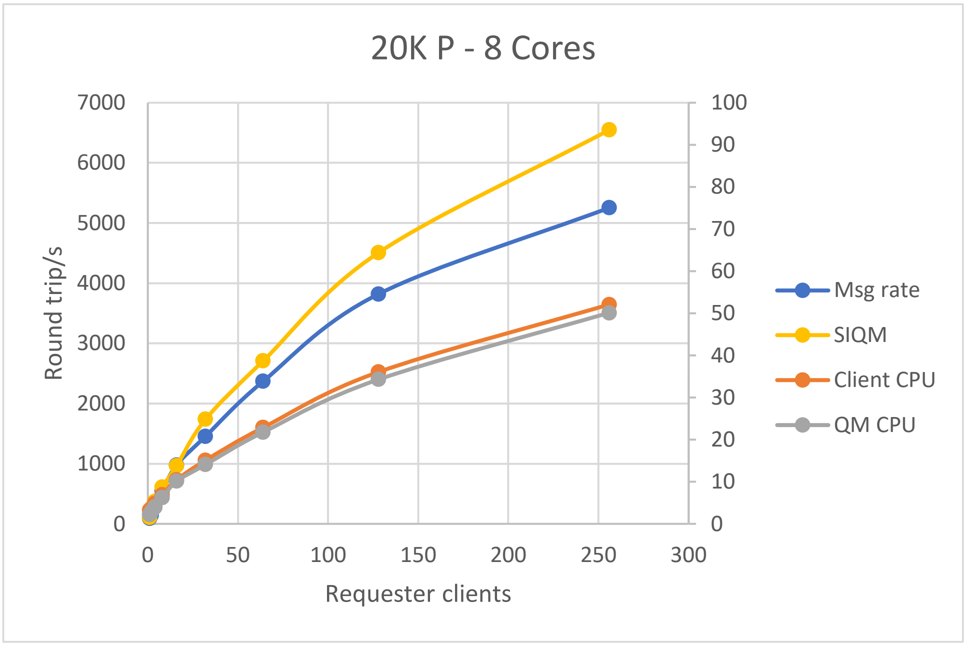


Figure 5 - 20K NHA Persistent messaging

The above chart shows that a small difference between SIQM and NHA performance appears as the message size has increased to 20KB. The NHA scenario still demonstrates over 5,000 round trip/s at 250 requester clients.

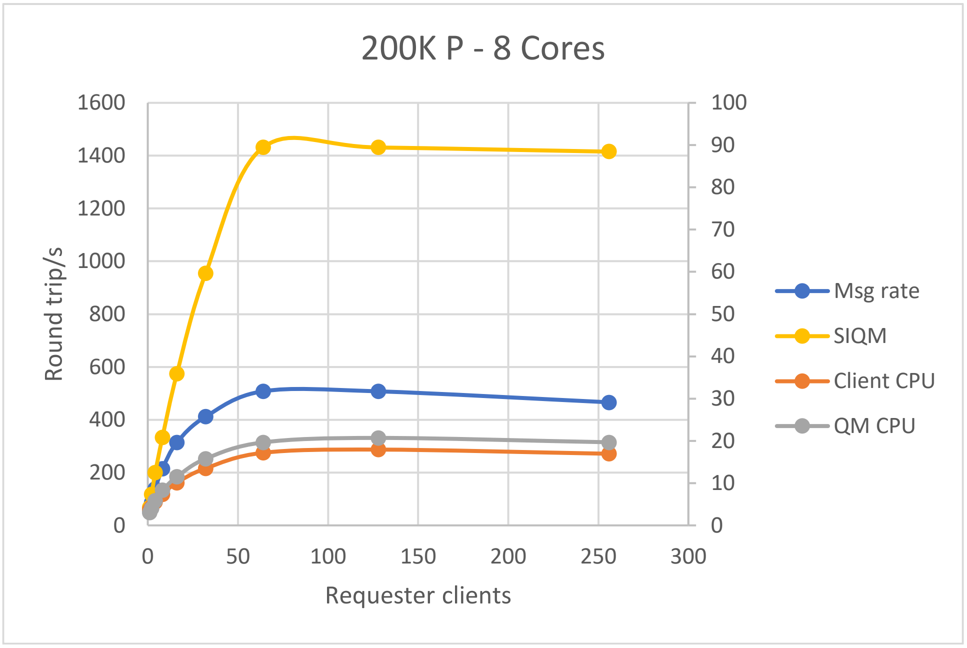


Figure 6 - 200K NHA Persistent messaging

At 200KB message size, the NHA scenario is limited by the 10GbE networking bandwidth between the nodes in different AZ. Higher specification EC2 instance types are available that could be utilized in such high bandwidth scenarios.

NHA:CRR Test configuration and results

This section uses the following test configuration:

- Persistent
- 2K / 20K / 200K messages size
- JSON formatted payload
- TLS12 enabled for Application, Live HA Group, Recovery HA Group and Live->HA replication
- 8GB PV allocated to QM instance
- 50GB PV allocated to Live group instances
- 100GB PV allocated to Recovery group instances
- IMGLOGLN set to 12500MB (25% of PV) – Controls when MQ media images are taken
- LZ4 Compression enabled, CompressionThreshold=64 within NHA groups and between the live and recovery groups
- AWS Network (between AZ): 5-10Gb
- AWS Network (cross region): 0.5-1Gb
- Core allocation to QM pod resources: 8 cores

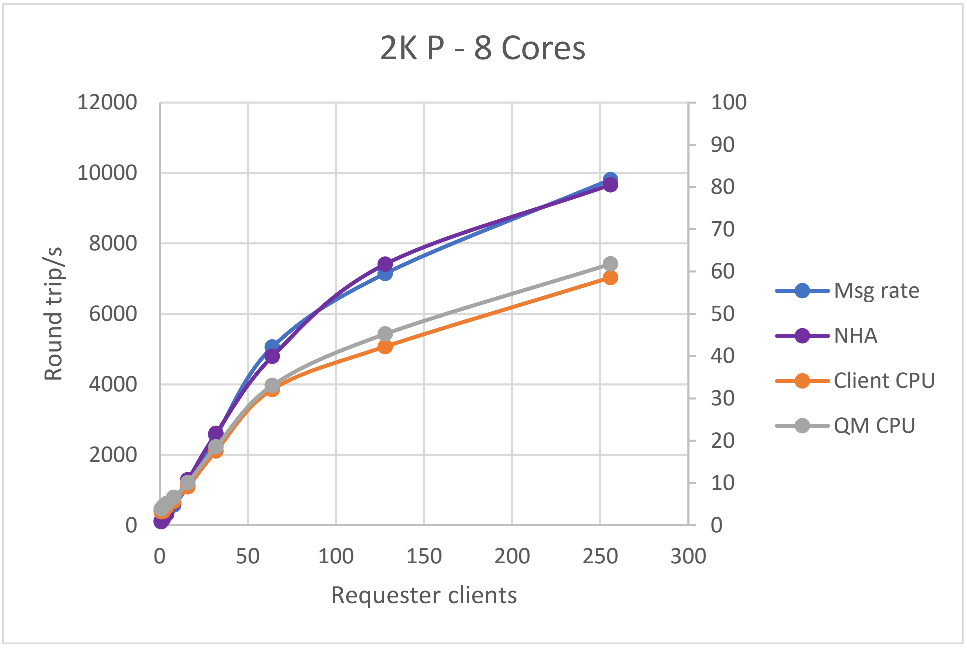


Figure 7 - 2K NHA:CRR Persistent messaging

The 2K test above performs identically whether you have CRR enabled to the DR Recovery group or not, even with 47ms of latency between the two regions. Hence, the blue NHA:CRR line is obscured by the purple NHA line above.

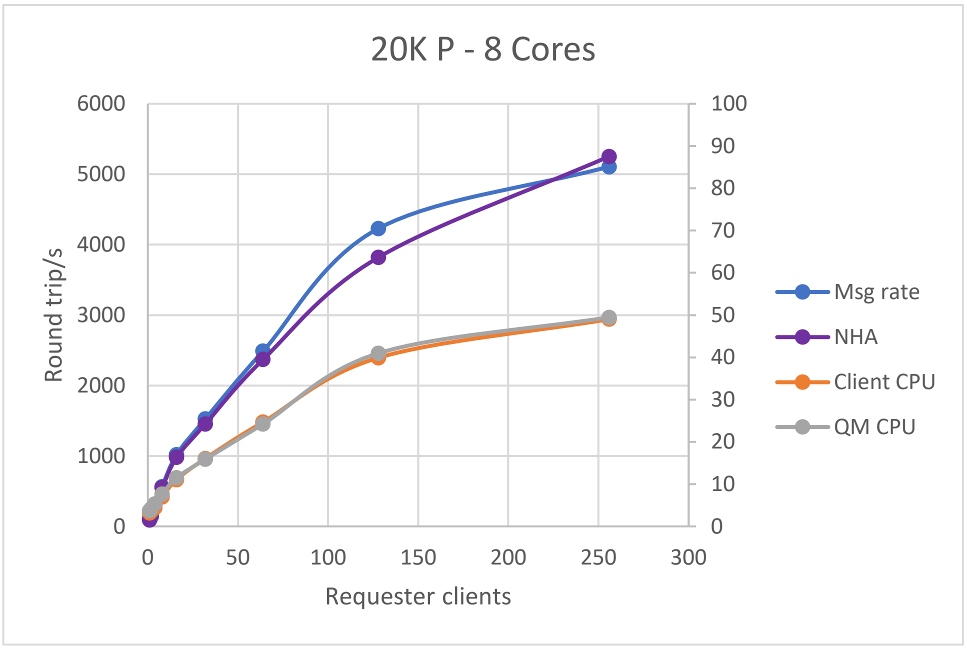


Figure 8 - 20K NHA:CRR Persistent messaging

The 20K test above again performs similarly whether you have CRR enabled to the DR Recovery group or not, even with the inter-region latency. A throughput rate of over 5000 round trip/s is achieved with a 20K message size; this is measured from the application perspective and does not reflect the message data that may be lost due to the asynchronous nature of the connection between the regions. Monitoring the RPA (Recovery Point Actual), with RPA being the amount of data that you would lose should a failure occur at the primary site, is an important part of deploying a distributed scenario. Our initial whitepaper on NHA:CRR discusses this aspect in more detail.

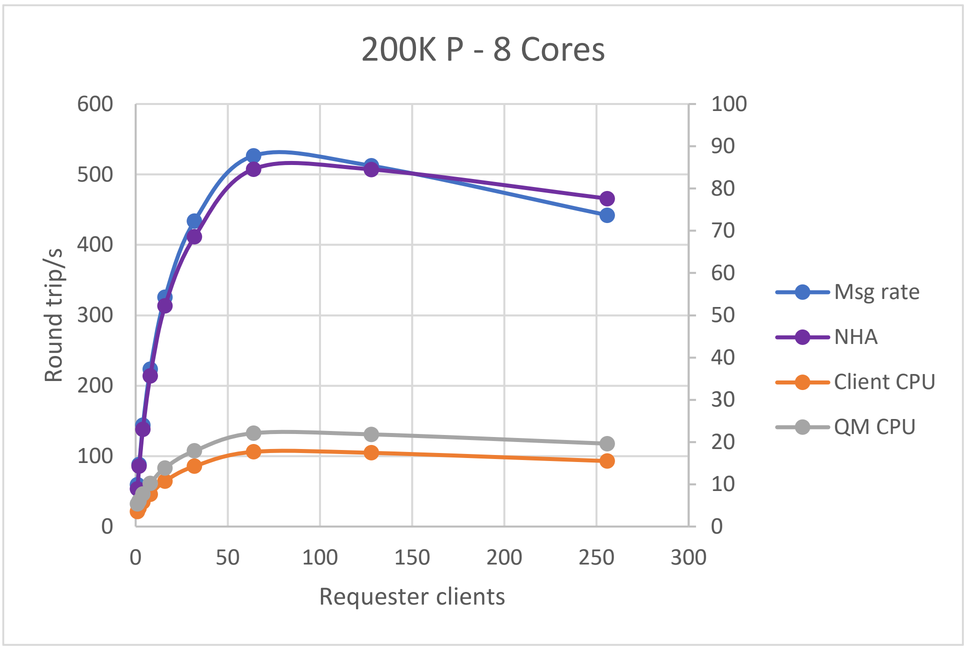


Figure 9 - 200K:CRR Persistent messaging

The above chart again shows that in this 200K distributed AWS scenario, the addition of the DR location has had little impact to the messaging rate that can be achieved by the highly available primary site.

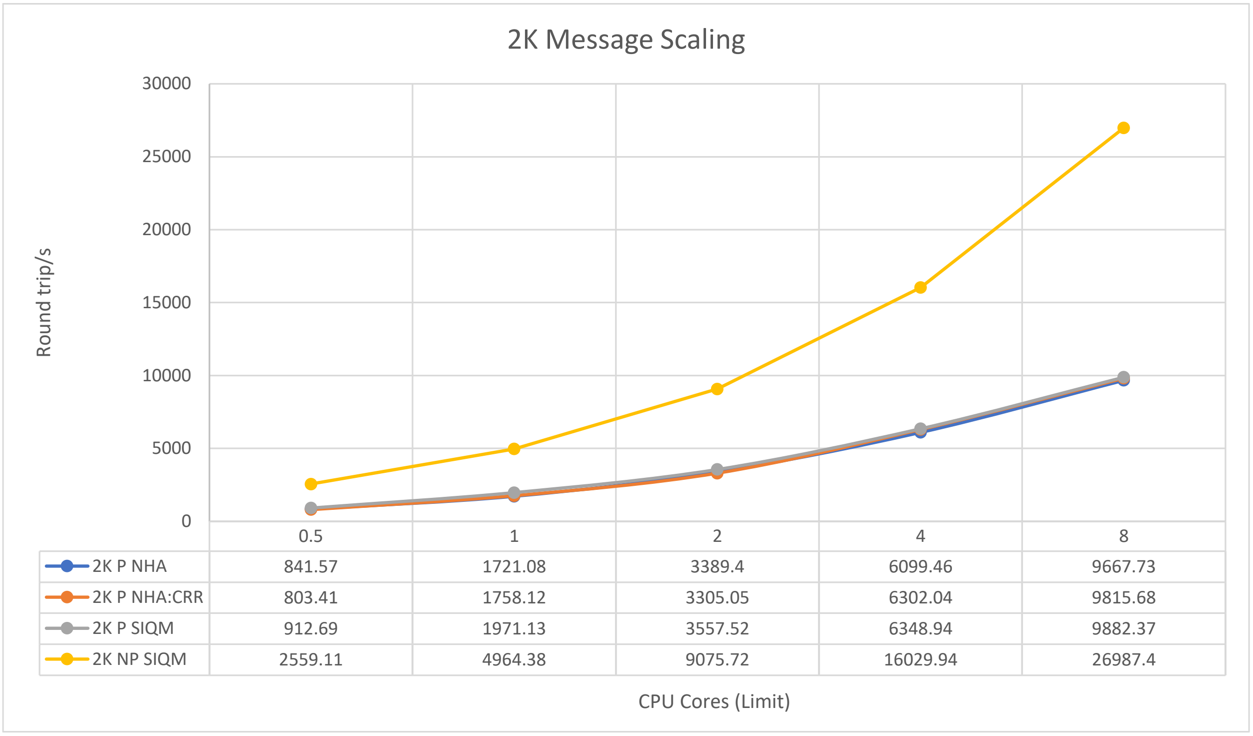
All this data shows that enabling NHA:CRR has very little impact when the replication infrastructure is able to support the messaging volume, and we have gained a full set of consistent data at our recovery site, albeit a little behind our live site. If the replication infrastructure cannot keep pace with the messaging workload at the live group, the recovery group may eventually require a rebase (i.e. a new recovery log to be sent to the recovery group).

Scaling Results

The results presented so far have been with a QM CPU request/limit of 8 cores. To illustrate how MQ performs in the AWS environment with smaller levels of CPU resources, the 2K, 20K and 200K message tests have been run against the MQ QM in multiple CPU configurations and the peak throughput noted.

The active and replica QM are each deployed with the same CPU request/limits.

The graph below shows how the MQ QM scales across varying CPU cores with a 2K message size.



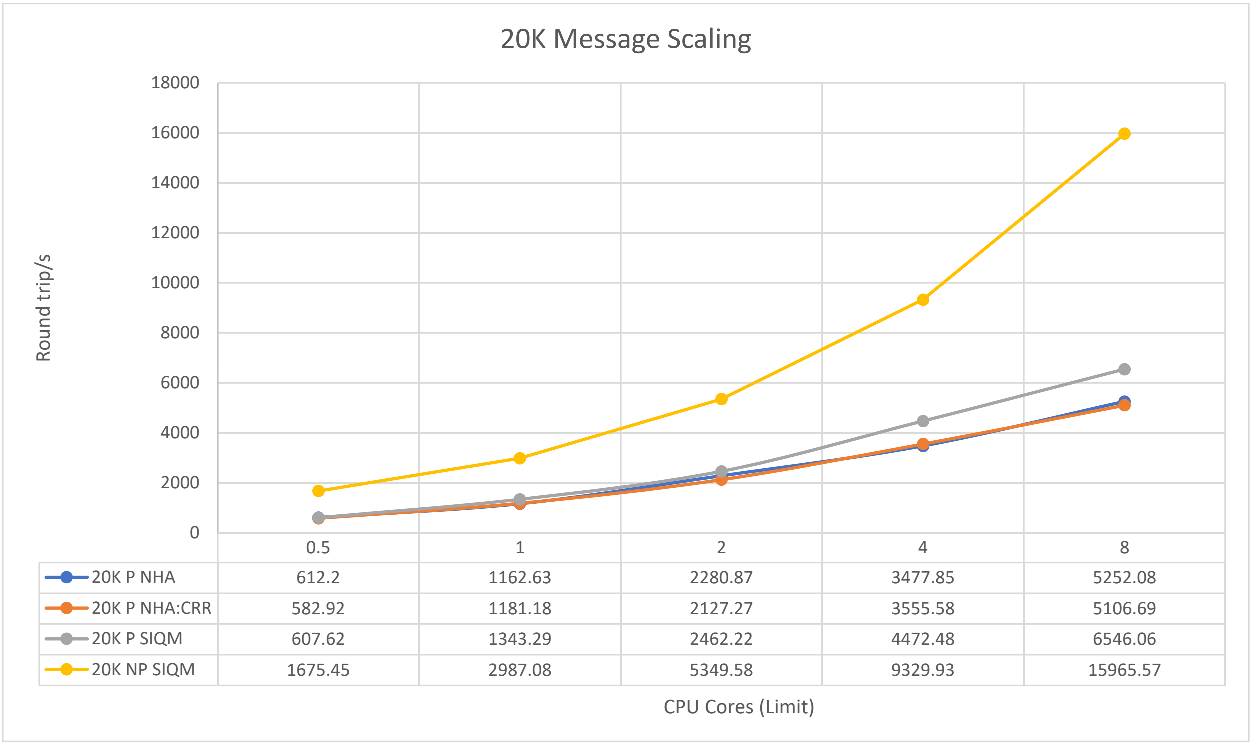
	0.5	1	2	4	8
2K P NHA	841.57	1721.08	3389.4	6099.46	9667.73
2K P NHA:CRR	803.41	1758.12	3305.05	6302.04	9815.68
2K P SIQM	912.69	1971.13	3557.52	6348.94	9882.37
2K NP SIQM	2559.11	4964.38	9075.72	16029.94	26987.4

Figure 10 - 2K Scaling

The chart above shows how we can support nearly 800 round trips/s at half a single CPU core right up to nearly 10,000 round trips/s at 8 CPU cores for Native HA Persistent messaging. The results are similar for SIQM and NHA:CRR scenarios.

For NP messaging against a SIQM the respective values are approximately 2,500 and 27,000 round trips/s.

The graph below shows how the MQ QM scales across varying CPU cores with a 20K message size.



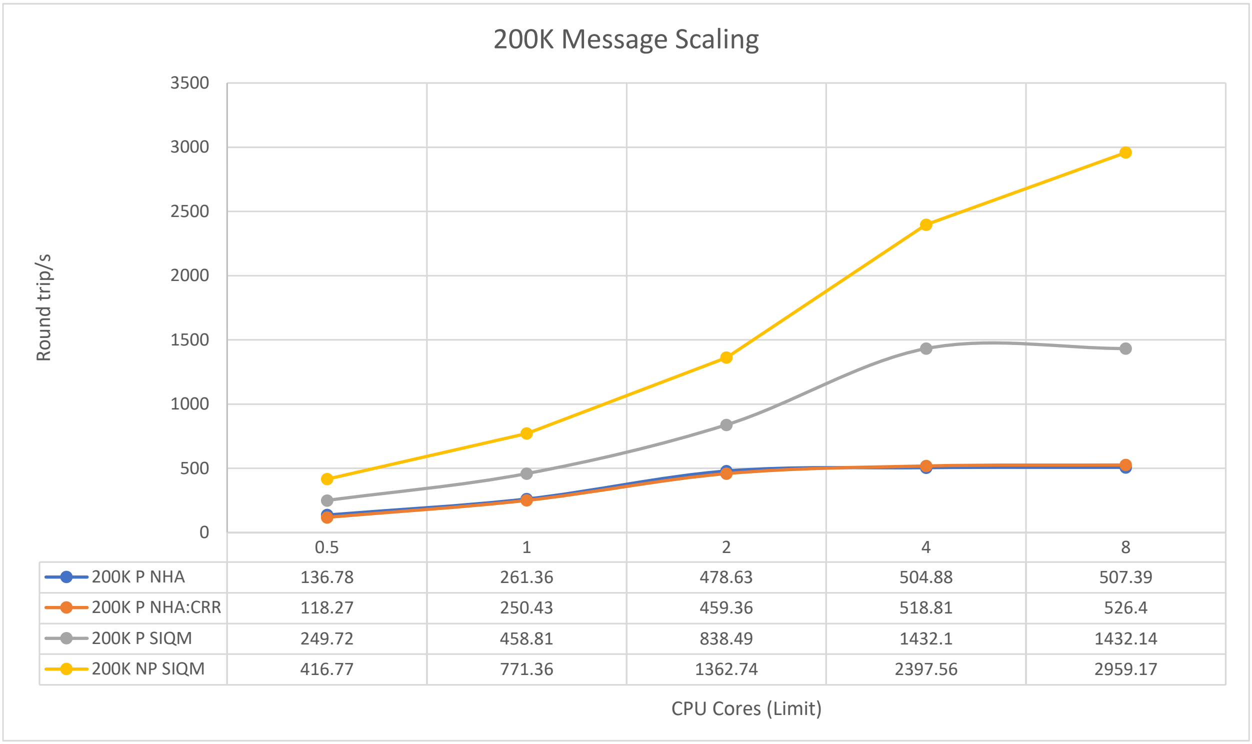
	0.5	1	2	4	8
20K P NHA	612.2	1162.63	2280.87	3477.85	5252.08
20K P NHA:CRR	582.92	1181.18	2127.27	3555.58	5106.69
20K P SIQM	607.62	1343.29	2462.22	4472.48	6546.06
20K NP SIQM	1675.45	2987.08	5349.58	9329.93	15965.57

Figure 11 - 20K Scaling

The chart above shows how we can support over 500 round trips/s at half a single CPU core right up to over 5,000 round trips/s at 8 CPU cores for Native HA Persistent messaging. The results are similar for NHA:CRR scenarios. The SIQM scenario starts to show a benefit over the NHA scenarios.

For NP messaging against a SIQM the respective values are approximately 1,500 and 16,000 round trips/s.

The graph below shows how the MQ QM scales across varying CPU cores with a 200K message size.



	0.5	1	2	4	8
200K P NHA	136.78	261.36	478.63	504.88	507.39
200K P NHA:CRR	118.27	250.43	459.36	518.81	526.4
200K P SIQM	249.72	458.81	838.49	1432.1	1432.14
200K NP SIQM	416.77	771.36	1362.74	2397.56	2959.17

Figure 12 - 200K Scaling

The chart above shows how we can support over 100 round trips/s at half a single CPU core right up to around 500 round trips/s at 8 CPU cores for Native HA Persistent messaging. The results are similar for NHA:CRR scenarios. The SIQM scenario shows a clear benefit over the NHA scenarios due to not having to send messaging data to replica QM

For NP messaging against a SIQM the respective values are approximately 400 and 3,000 round trips/s.

Conclusion

The new capabilities of NHA:CRR provide further availability to your messaging infrastructure by supporting asynchronous replication to a disaster recovery (DR) site remote from the location of your Live HA group (already providing High Availability). The AWS combination of multiple AZs and regions helps create a truly available solution for MQ.

AWS offers a wide choice of deployment options for MQ on OpenShift scenarios, with a myriad of compute nodes, storage options, network bandwidth, AZs and regions available for use. This whitepaper illustrates the performance of some popular AWS choices to help guide your selection. As always, you are recommended to test with your own application clients and deployment choices to ensure your performance requirements can be met.

Links

cphtestp
https://github.com/ibm-messaging/cphtestp

Performance Test on NHA:CRR
https://ibm-messaging.github.io/mqperf/PerformanceTestOnNHACRR.pdf

Native HA Performance on OpenShift
https://github.com/ibm-messaging/mqperf/blob/gh-pages/openshift/OCP4.12-NativeHA.pdf

NHA:CRR Documentation
https://www.ibm.com/docs/en/ibm-mq/9.4?topic=containers-native-ha-cross-region-replication

NHA:CRR Advanced tuning
https://www.ibm.com/docs/en/ibm-mq/9.4?topic=operator-advanced-tuning-native-ha-crr

Monitoring system resource using amqsrua
https://www.ibm.com/docs/en/ibm-mq/9.4?topic=stmat-monitoring-system-resource-usage-by-using-amqsrua-command

Metrics published on the system topics
https://www.ibm.com/docs/en/ibm-mq/9.4?topic=trace-metrics-published-system-topics