# AMS and SSL/TLS Performance on the IBM MQ Appliance

## Objective

Examine the performance of AMS(Advanced Message Security) and SSL/TLS(Secure Sockets Layer/Transport Layer Security) on the MQ Appliance, including the new AMS Quality of Protection Mode (Confidentiality).

## Background

IBM MQ customers often require message payloads to be encrypted while in transit and at rest to comply with various security mandates. TLS protects message data in transit, whilst AMS protects message data in transit and at rest.

IBM MQ V9 delivered a new AMS Quality of Protection called 'Confidentiality' which utilizes symmetric keys without message signing to provide a faster way to allow messages to be transferred securely and protect their payload data at rest.

TLS is used to encrypt the data conversation sent/received between the client and queue manager to provide link level security. AMS encrypts each individual message and is only decrypted at the receiving application. Once a non-AMS message is received over a TLS channel, the message is stored in plaintext in both QM(Queue Manager) memory and in the queue log/data files.

Both AMS and TLS provide the user with different options to protect the message data being flowed. Each option can affect the overall messaging performance of the scenario and this paper will illustrate the performance impact of those options.

## Scenario

A couple of different scenarios will be used to analyze the AMS and TLS Performance:

- Single queue – All clients send and receive from a single queue
- Multiple queue – All clients send and receive from one of 10 queues

For this investigation, unless stated otherwise a 2KB persistent message is used in all tests. The messaging scenario is a request responder scenario as featured in the current distributed and appliance performance reports.

The following encryption details apply respectively to each of the scenarios:

- AMS
  - Each message is encrypted for a single recipient
  - The signing algorithm used in the Integrity and Privacy modes is SHA512. The encryption algorithm used for the Privacy and Confidential modes is AES256
- TLS
  - The TLS_RSA_WITH_AES_256_GCM_SHA384 CipherSpec will be used. The signature algorithm is SHA256WithRSA.

A comparison will be made between all of the AMS Quality of Protection settings:

- None – No signing or encryption applied
- Integrity – Message is signed
- Privacy – Message is signed and encrypted
- Confidentiality – Message is encrypted and key may be reused
    - Key reuse setting of 32 will be used

The key reuse setting controls how often the symmetric key that is regenerated. For more information on this and additional detail on the Qualities of protection, please see the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q127085_.htm

For the comparisons in this whitepaper, an AMS Confidentiality key reuse value of 32 was chosen; this is considered a reasonable compromise in the tradeoff between the costs of key renegotiation (i.e. scenario performance) and security. A set of data collected that demonstrates how the performance linearly varies with the key reuse configuration was demonstrated in the AMS report published for the distributed platforms: http://ibm-messaging.github.io/mqperf/AMS.pdf

AMS Confidentiality(32) provides 94% of the performance of AMS Confidential(Unlimited), but will only reuse the same symmetric key for 32 messages.

## Environment

These tests use 2 x86_64 Linux servers (see Appendix A for their specification); Server 1 hosts the requester clients, the MQ Appliance hosts the QM under test and Server 2 hosts the responder applications.

When providing a message buffer to the MQGET API to receive your AMS message, ensure the buffer is larger than the expected message size, as the encrypted payload size is larger than the original message length. For the tests featured in this report a 20KB buffer was supplied, although on subsequent analysis, a 4K byte buffer would have been sufficient as largest buffer required was 3406 bytes for the signed and encrypted data in an AMS Privacy message(original message size was 2048 bytes).

The version of MQ used in these tests is MQ V9.0.2.

# Results

## Single queue

The graph below shows the results from the single queue test:

Note that the CPU represented on the charts in this whitepaper is an average of the two client machines featured in the test; this is of more interest than the server CPU (which is traditionally featured on such graphs) as the costs of encryption and decryption are handled by the client machines.

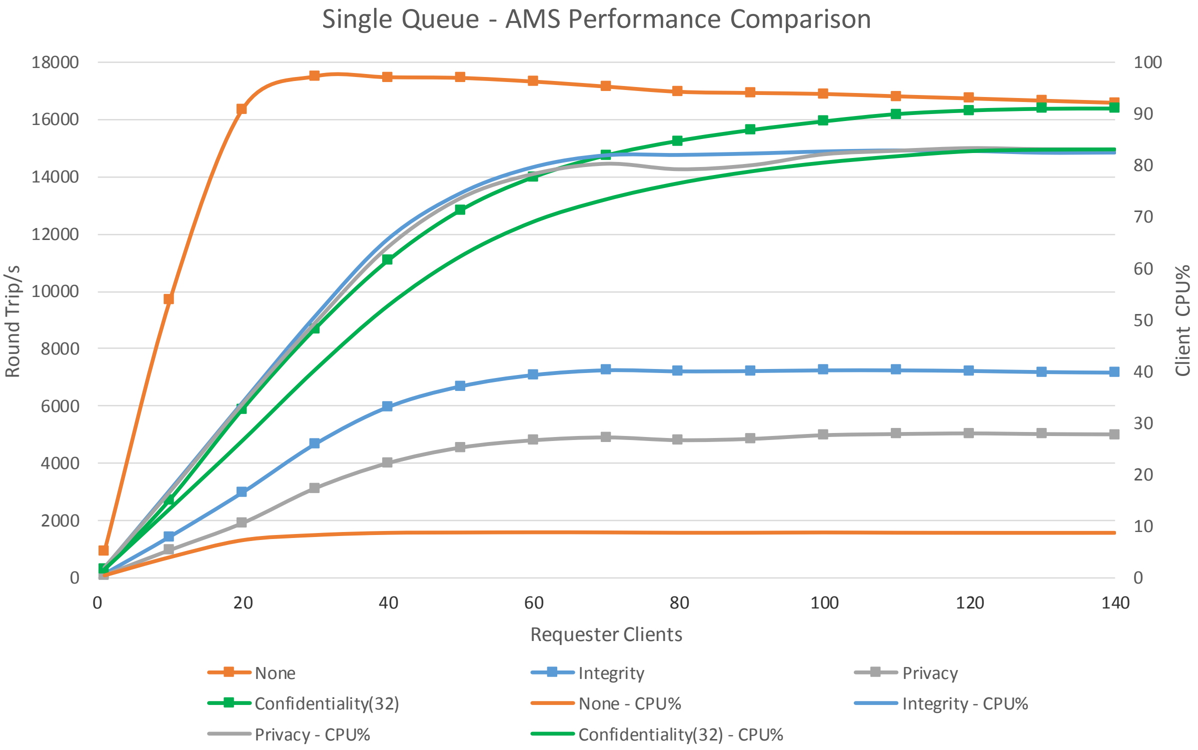

*Figure 1 - Single queue AMS Comparison*

The performance of AMS Confidentiality is more than 3x faster than the AMS Privacy mode. At 140 clients, the performance of AMS Confidentiality was almost identical to than when not using AMS (although significantly more CPU was being expended by the clients).

### Multiple queue

The graph below shows the results from the multiple queue test:

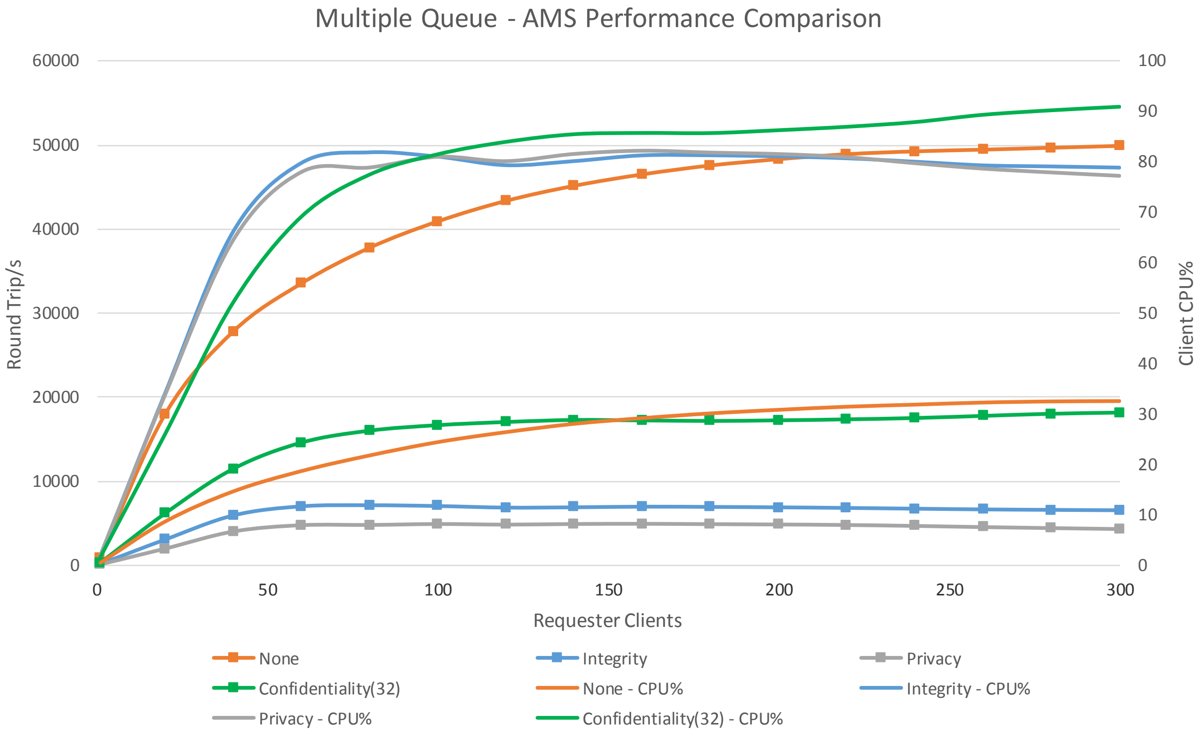

*Figure 2 - Multiple Queue AMS Comparison*

Adding multiple queues into the scenario has increased the performance of AMS Confidentiality(32) to over 4x faster than the AMS Privacy mode. It has also increased the performance of the non AMS scenario (by relieving queue lock) and at 100 clients, the performance of AMS Confidentiality(32) was just under half of the performance when compared with not using AMS.

The peak throughput achieved for the AMS Confidentiality(32) measurement at 300 clients was just over 18,000 round trip/s. The request/responder scenario utilizes a request and a reply queue, so for each round trip, 2 message put and 2 message get operations take place. For a single put/single get scenario, the peak performance that you might obtain in the same environment is over 36,000 msg/sec.

### Effect of poorly sized receive buffer

If a buffer that is not adequately sized for performing the MQ Get operation, poor performance can result from the MQ client having to perform multiple message retrievals from the MQ QM. The graph below takes the AMS Confidentiality(32) result from the multiple queue scenario and compares it with the result if a message buffer of just 2K had been provided:

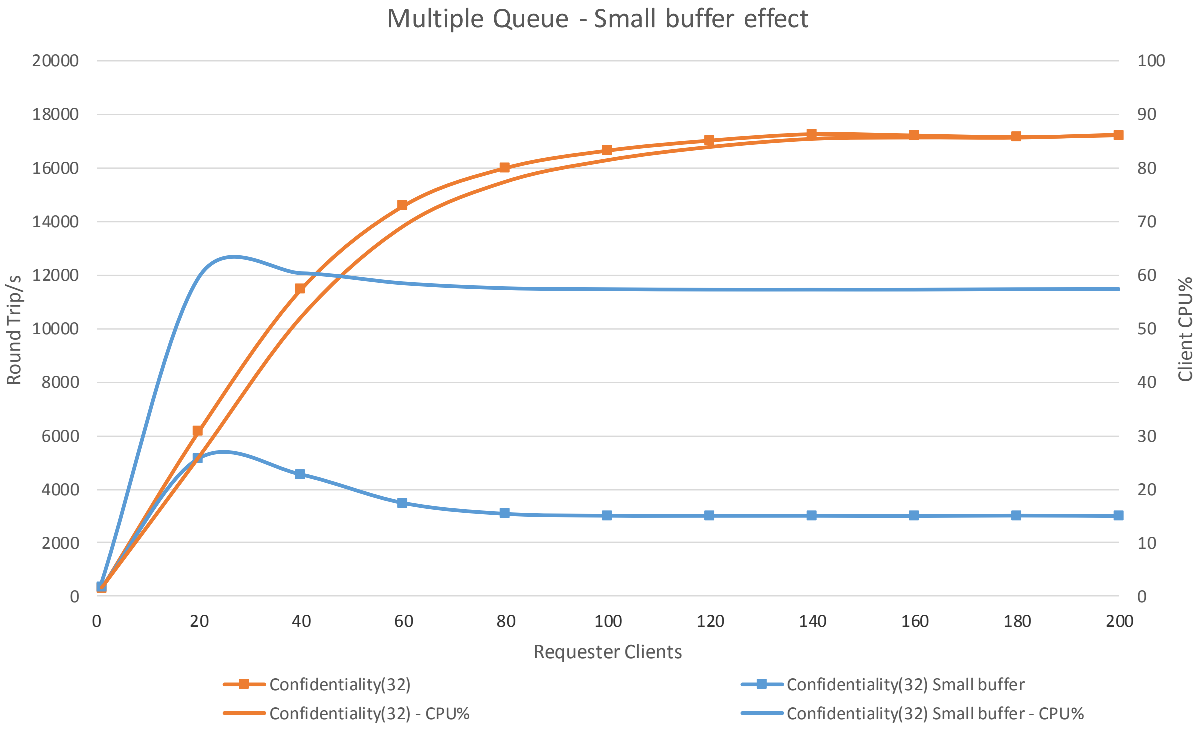

*Figure 3 - Effect of small receive buffer*

### Comparison with TLS

In the graph below, we compare the performance of AMS Confidentiality(32) with TLS. Although both offer on-the-wire encryption, AMS also protects the message at rest at the QM. 3 variants of the TLS scenario have been performed with different values for the SSLKeyResetCount. The SSLKeyResetCount controls how much data flows between the client and the server before the TLS secret key is renegotiated. A reasonable comparison would be the 64K scenario as that would offer similar protection (amount of data transmitted between each key renegotiation) as AMS Confidential(32). Performance data with the SSLKeyResetCount set to 256K and 0 (never renegotiate) have also been included.

The server CPU is presented (rather than the Client CPU as used in previous graphs) to illustrate the comparative server CPU utilization between AMS and TLS.

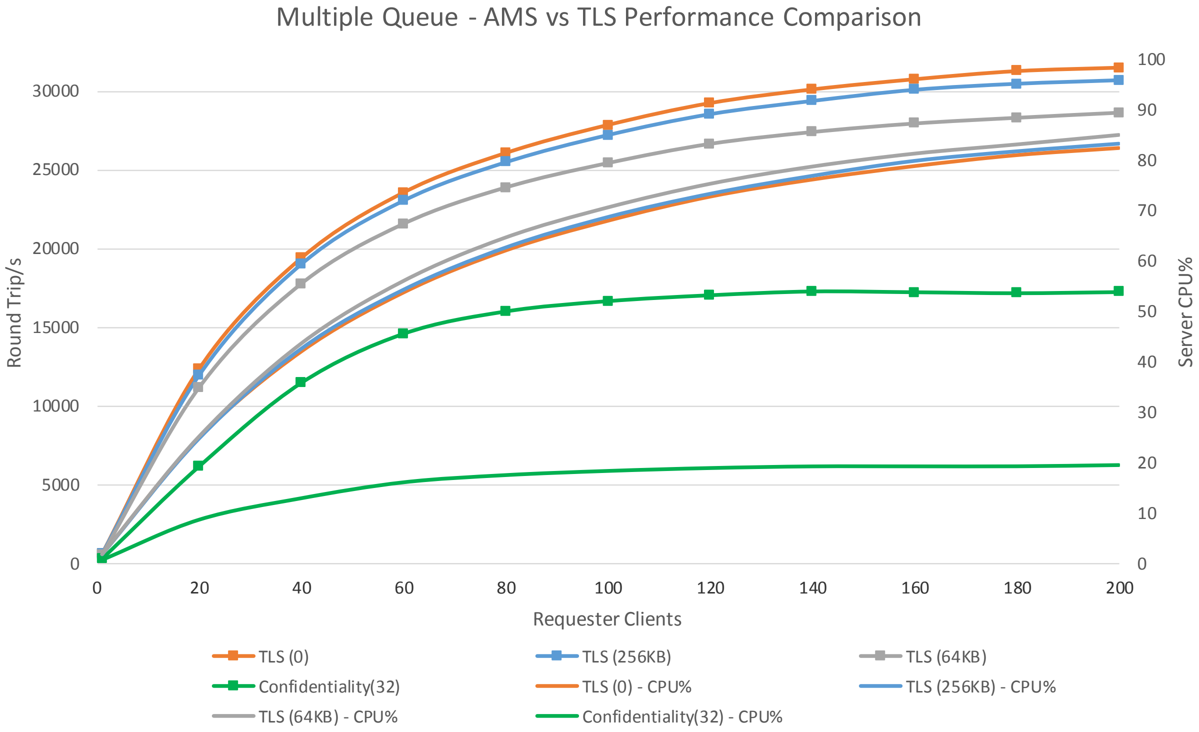

*Figure 4 - AMS vs TLS Performance Comparison*

This graph shows that in this scenario, TLS will outperform AMS in terms of throughput but will also use a much larger slice of your Server CPU. Conversely the multiple responders per queue competing for the same messages causes additional CPU to be utilized by the AMS clients.

If you convert the throughput rate into an internal transaction rate of the server ((Throughput/Server CPU)*100), AMS is much more efficient because the encryption is being handled by the clients:

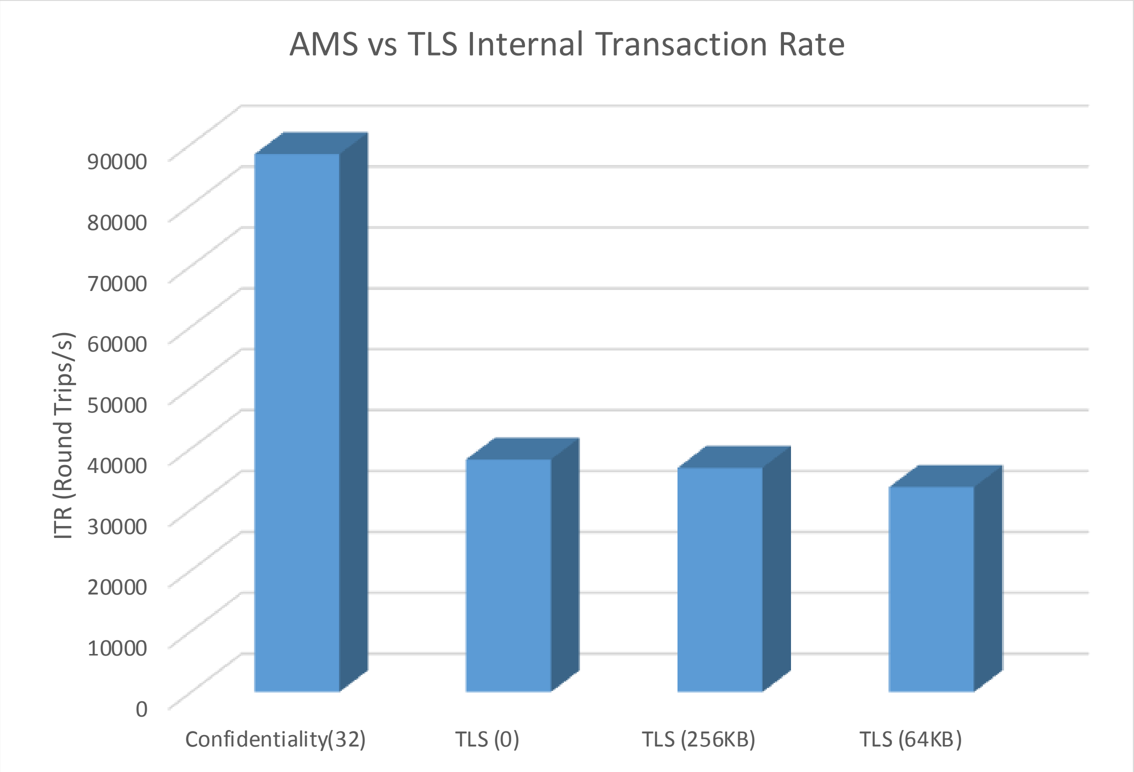

*Figure 5 - AMS vs TLS Internal Transaction Rate*

### Optimizing AMS Performance

In the previous sections the AMS results have used multiple queues (10) over which the workload has been distributed. One optimization that can be used (if applicable to your scenario) is to use more queues and have dedicated producers/consumers per queue. The following chart compares the various encryption scenarios already explored (now using Non Persistent messages), with up to 200 requester threads across 200 queues with 200 responder threads.

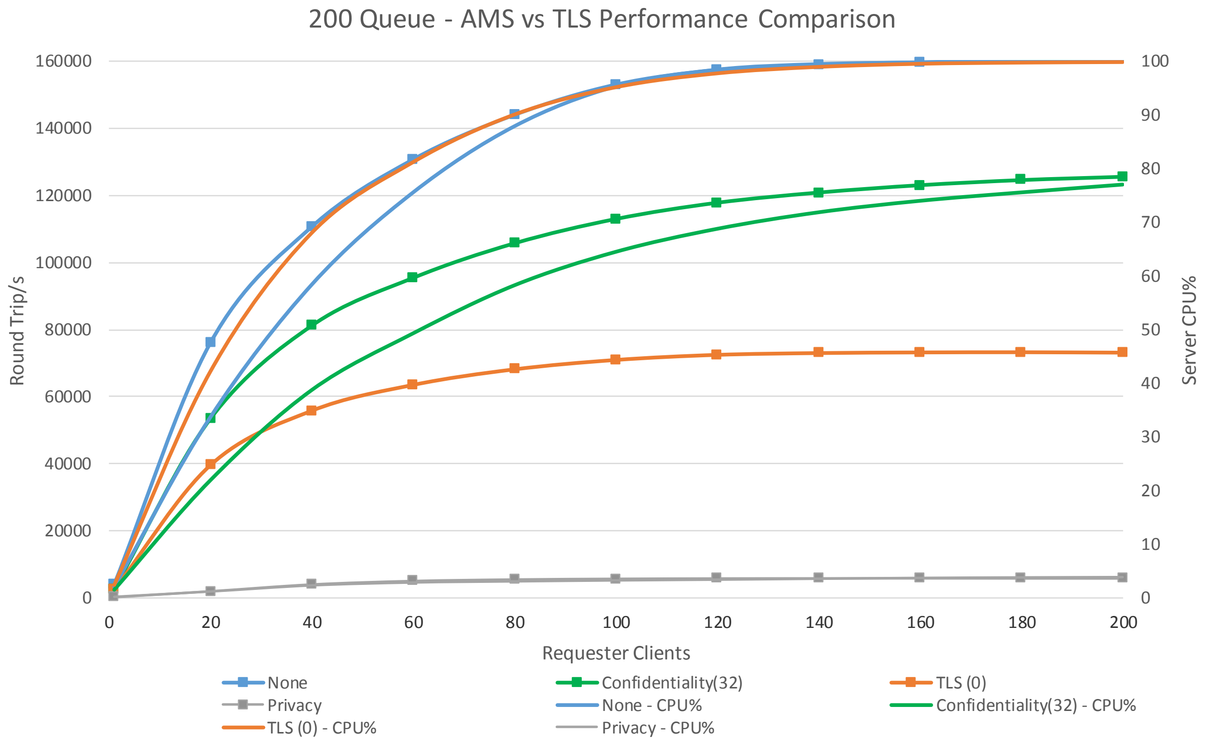

*Figure 6 - AMS vs TLS NP Performance Comparison*

In this scenario you can see that by avoiding contention by the AMS clients, the throughput is improved such that AMS(32) is now 72% faster than TLS(0) and is now limited by the CPU on the AMS clients. AMS(32) is now over 20x faster than AMS Privacy mode.

## Conclusions

The new AMS Confidentiality mode provides end to end security for your message payload so that it is protected in transit and whilst at rest in the Queue Managers filesystem within the MQ Appliance. It reduces asymmetric key encryption and thus can offer faster performance than previously supported AMS policies. It also allows the user to define how often to reuse the same symmetric key for payload encryption, thus providing flexibility in the choice between key regeneration frequency and performance. AMS Confidentiality mode now outperforms TLS in some scenarios whilst also reducing CPU utilization at the server.

## Author

The author of this whitepaper is Sam Massey who works in the MQ Performance Team at the IBM UK Laboratory, Hursley. If you have any questions or comments on this paper, please contact him at smassey@uk.ibm.com

## Appendix A

The three machines used for the performance tests in this report have the following specification:

| Category | Value |
|---|---|
| Machine | x3550 M5 |
| OS | Red Hat Enterprise Linux Server 7.3 |
| CPU | 2x12 (2.6Ghz) |
| RAM | 128GB RAM |
| Network | 10Gb/40Gb Ethernet |
| Disks | 2x 480GB SSD |
| RAID | ServeRAID M5210 (4GB Flash RAID cache) |