Abhishek Suresh

CSCI 5817 – Database Systems

Project 1 – Measuring a Cassandra Cluster

Collaborators: Swathi Upadhyaya, Suma Dodmani

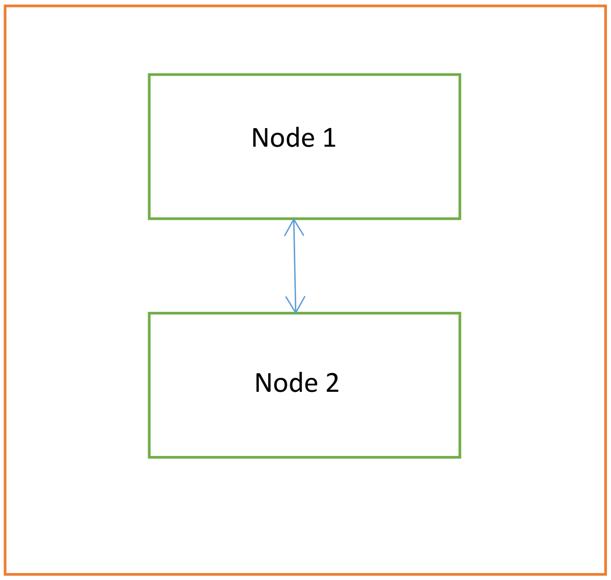
Introduction

Apache Cassandra is an open-source distributed NoSQL database management system designed to handle large amounts of data. Cassandra is a highly scalable NoSQL system, which is sometimes subject to high read and write latencies. The purpose of this project was to measure a Cassandra cluster, first by measuring simple read and write latencies, and then by measuring clock skew. Clock skew is measured by passing messages between a pair of nodes and measuring their latencies, and calculating the clock offset or the difference between the local times of the two nodes (after the two nodes have been synchronized with an NTP server).

These measurements were taken over three topologies. In the first, the Cassandra nodes were located inside of Docker containers on a single host virtual machine. In the second, each node was located on a separate virtual machine in a single region with a separate NTP server virtual machine synchronizing time with each node. In the third topology, the nodes were spread out across different regions. The details of these topologies and the results of the measurements are as follows. The topologies were set up using Docker containers and Amazon AWS Ubuntu 16.04 t2.micro instances as virtual machines.

First Topology

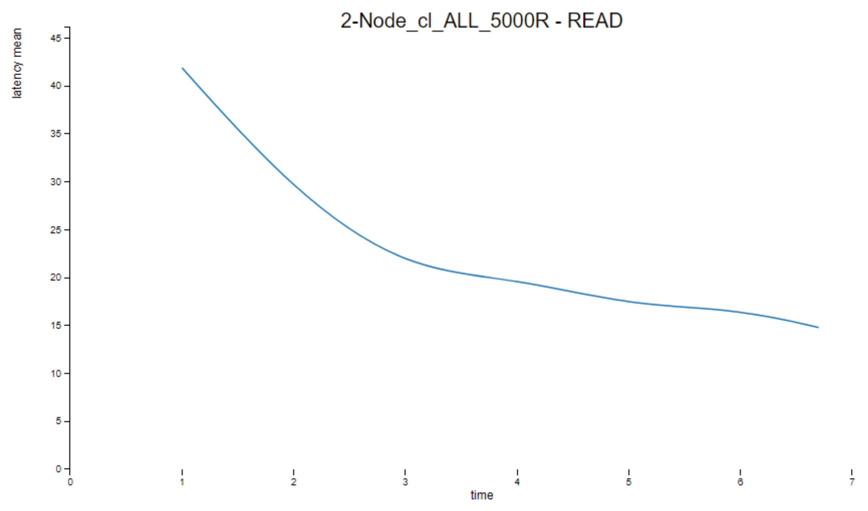
In this topology, each of the nodes in the cluster was located inside a Docker container (represented by green boxes) and both these containers were located inside of a single host virtual machine (represented by the orange box).



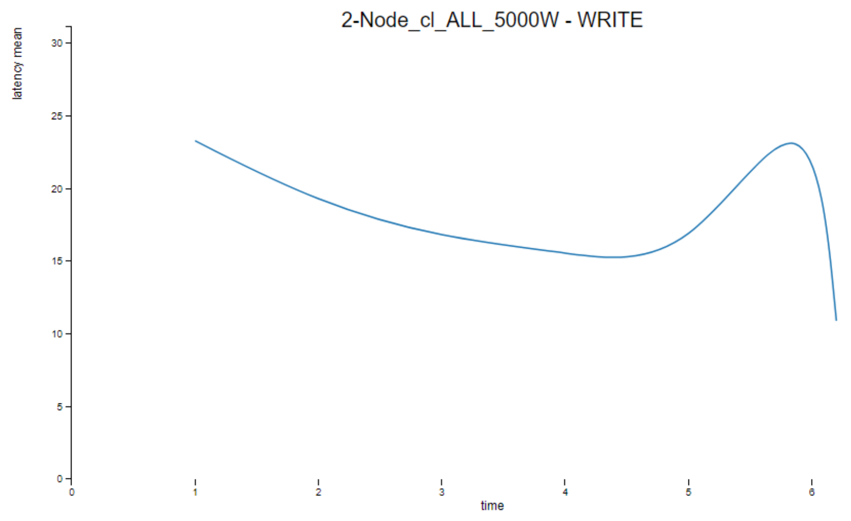
Location: Ohio

For read and write latencies, 5000 reads/writes and 16 threads were given. In this topology, the mean of read latency is observed to be about 19.9 ms, and the mean of write latency was about 17.7 ms. Cassandra's read operations are generally much slower than writes, because reads involve more I/O. The spike in write latency towards the end can be caused by things like network issues or changes in usage patterns, although in this case the usage did not really change much.

Read latency:

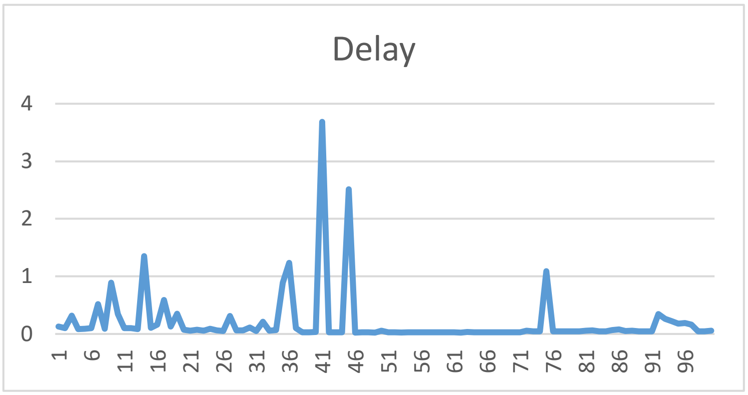


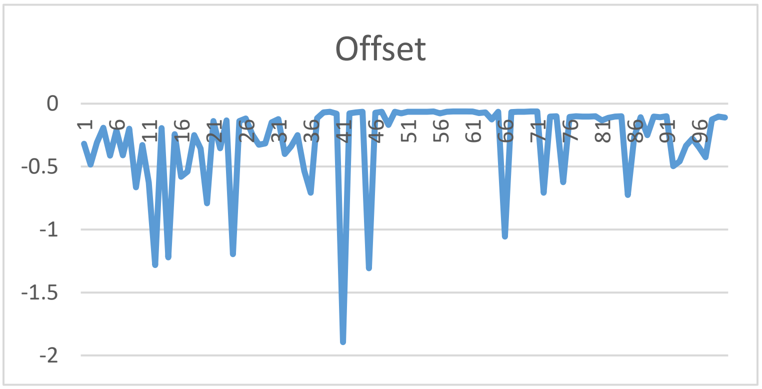
Write latency:



The mean offset observed in this topology was -0.29 s. This is a relatively small number which makes sense given that both nodes were located on the same virtual machine.

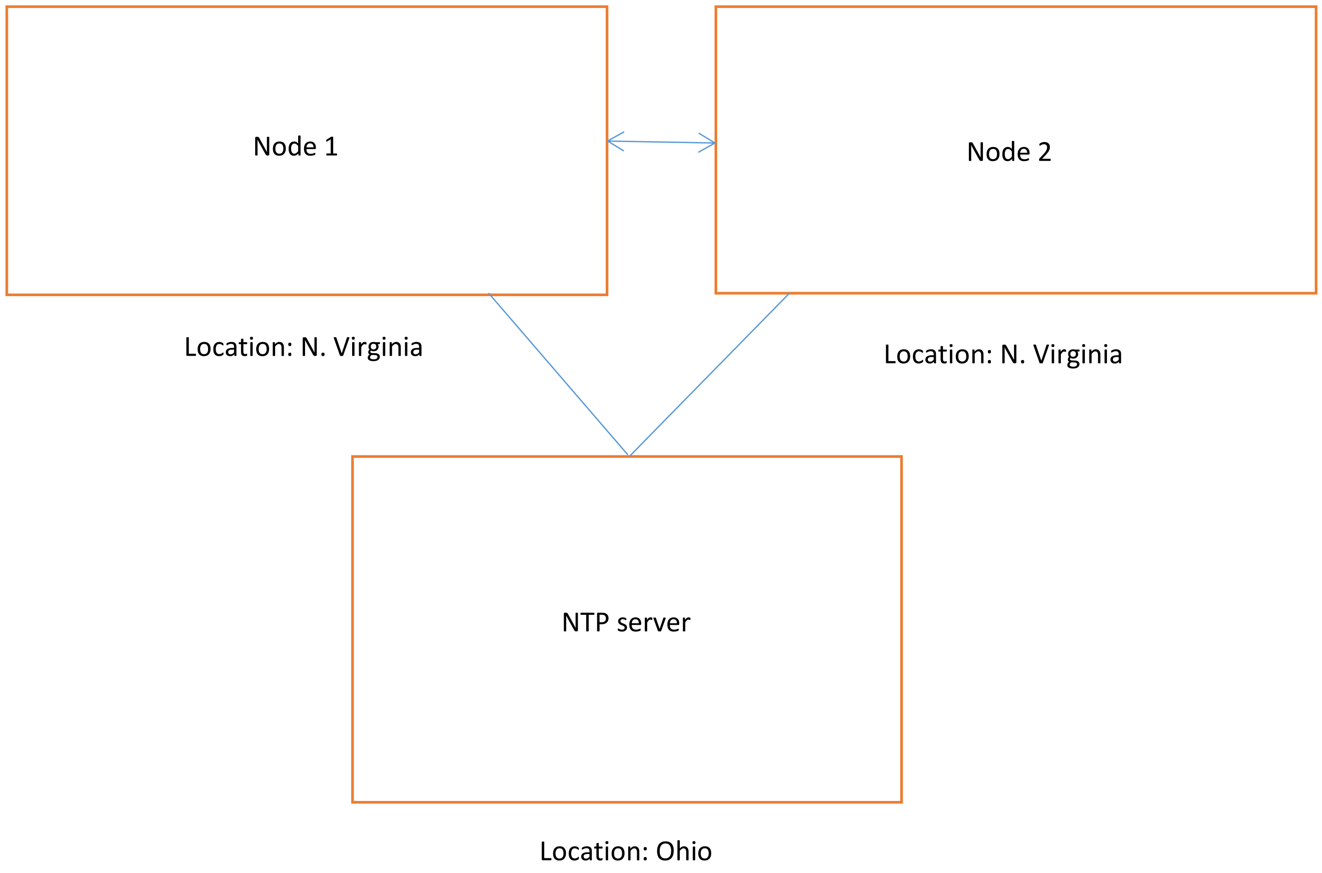
Clock skew:





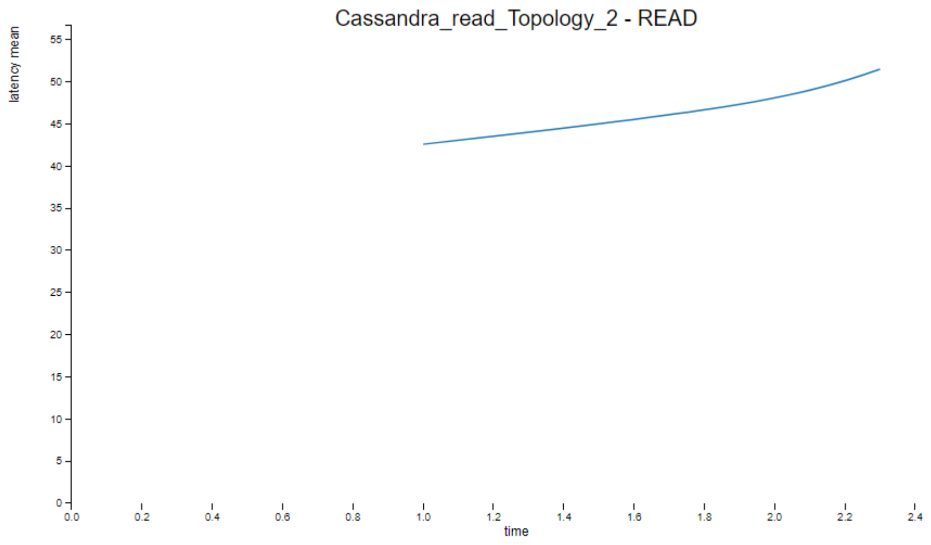
Second Topology

In the second topology, each node was located on a separate virtual machine in the same region and all the nodes were connected to one another in a cluster. There were two virtual machines containing nodes, both located in Northern Virginia. Each node received time from a virtual machine that acted as an NTP server, located in Ohio.

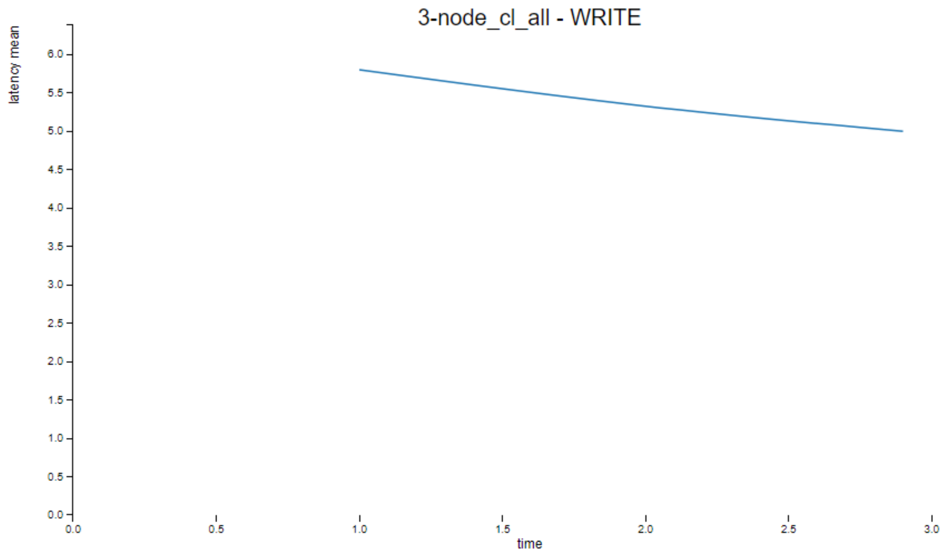


As for the first topology, 5000 reads/writes were given. The mean read latency was observed to be 47.7 ms and the mean write latency was 5.3 ms. Read latency is usually higher than write latency in Cassandra, but the write latency was extremely low as compared to the first topology. Amount of memory available is a factor that affects write performance, along with things like the replication factor, connection speed and commit log/data file location. The larger amount of memory available to each of the nodes in this topology may have advantaged the write performance greatly,

Read latency:

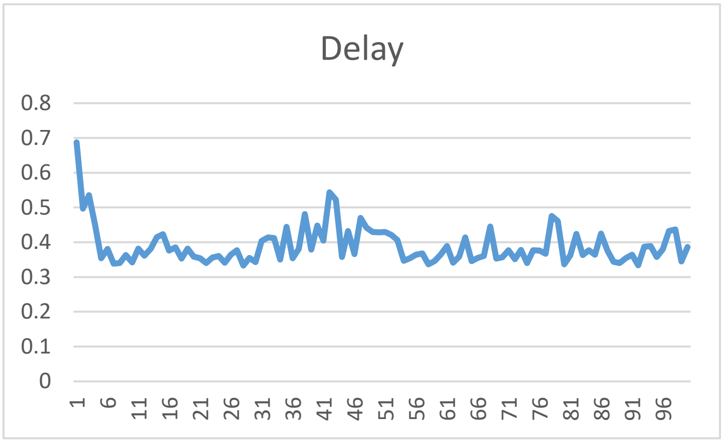


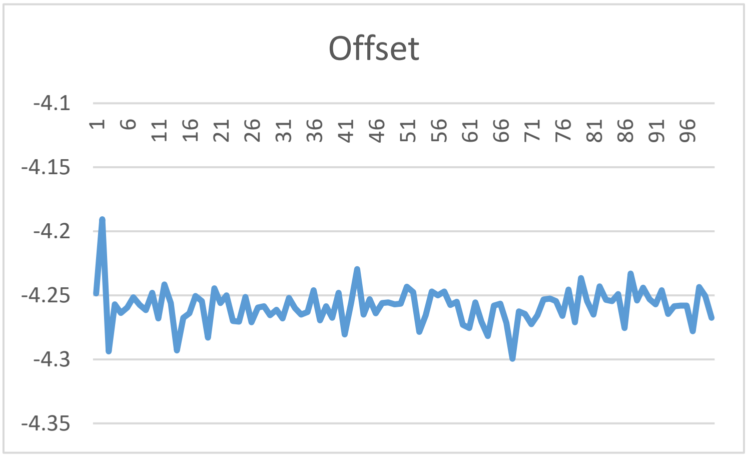
Write latency:



The mean offset obtained for this topology was -4.26 s. However, this was the value obtained after the cluster was up and running for a few hours. The initial mean offset was -1.02 s. Giving the cluster time seemed to cause higher offset.

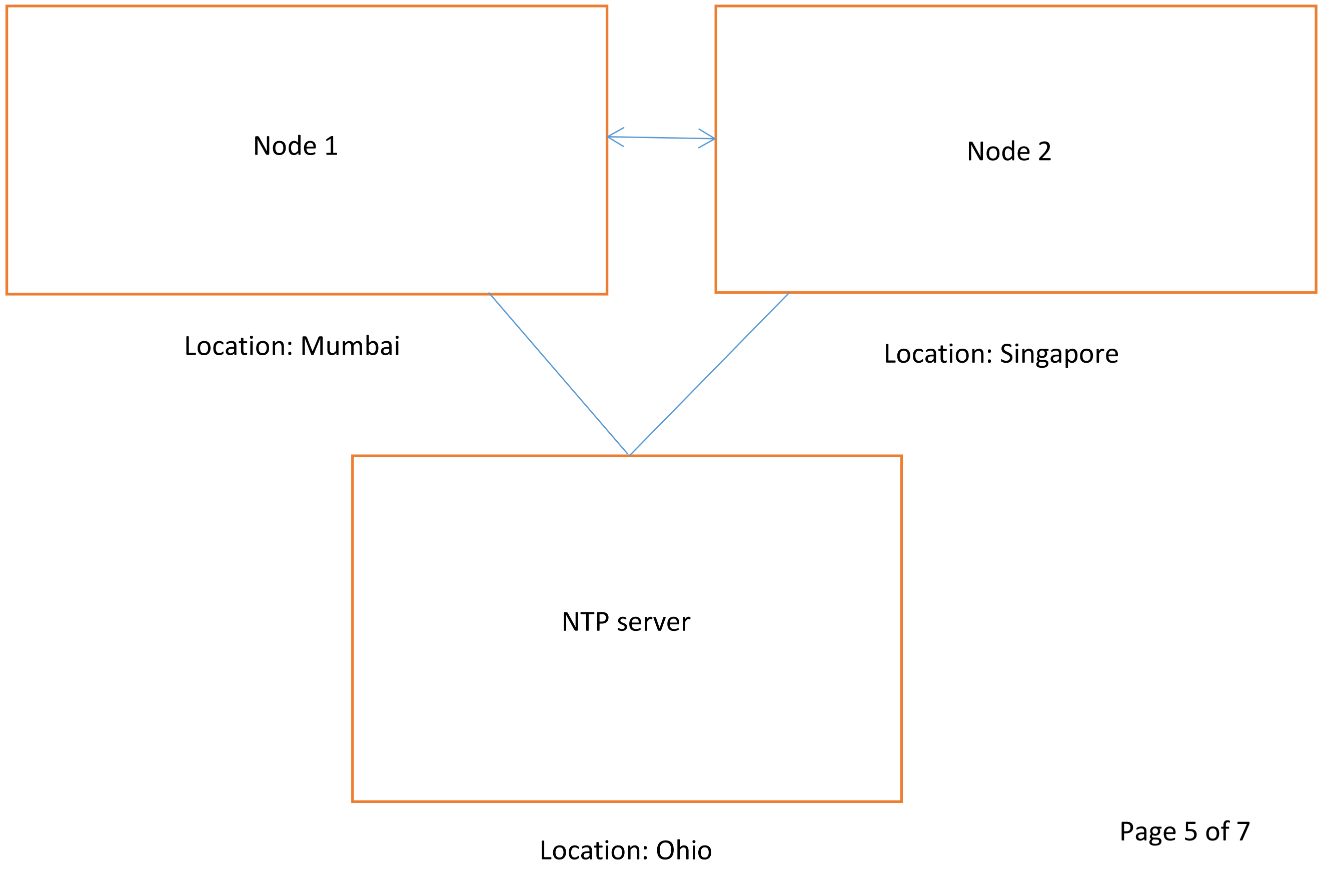
Clock skew:





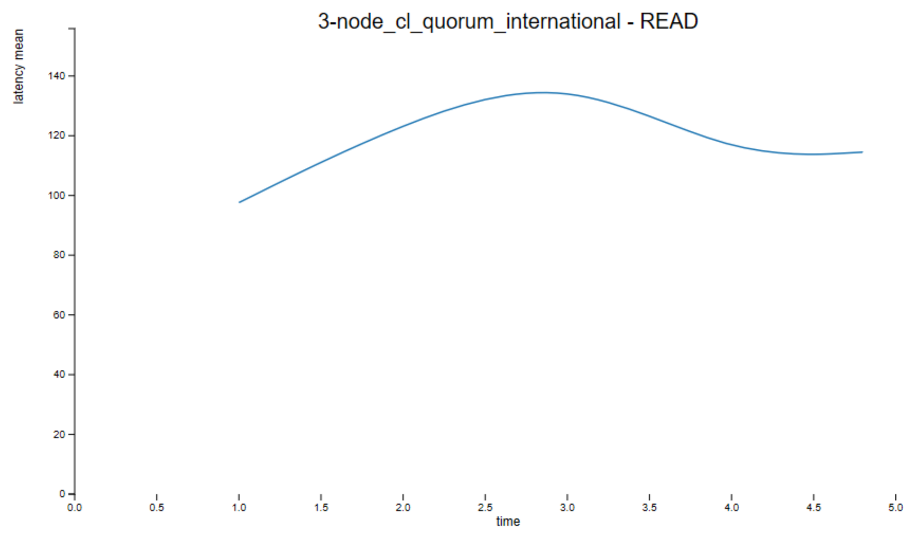
Third Topology

This topology was similar to the second topology, except that the nodes were located in different regions, one in Mumbai (Asia-Pacific South) and another in Singapore (Asia-Pacific Southeast).

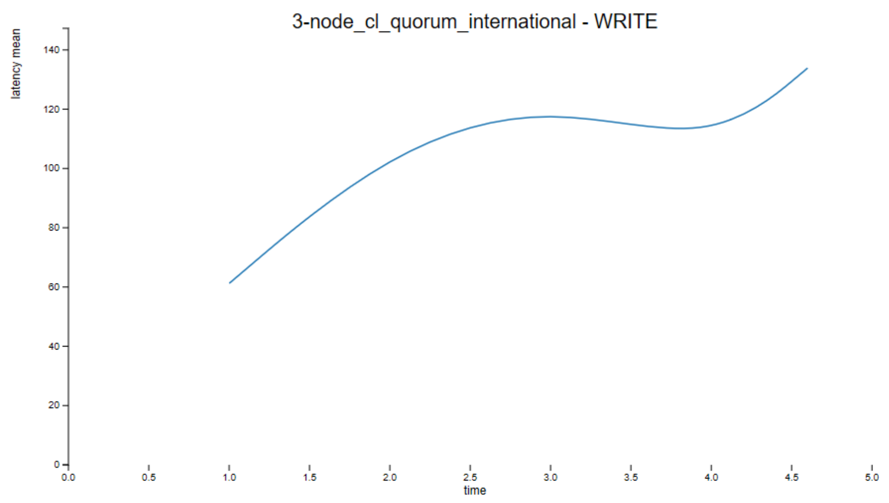


For this topology, it was not possible to give very many reads or writes on the cassandra-stress tool as large numbers caused the nodes to go down. This may be due to the instability or memory shortcomings of the AWS instances that were used. For 500 reads and writes, the mean read latency obtained was 122.1 ms and the mean write latency obtained was 114.7 ms.

Read latency:

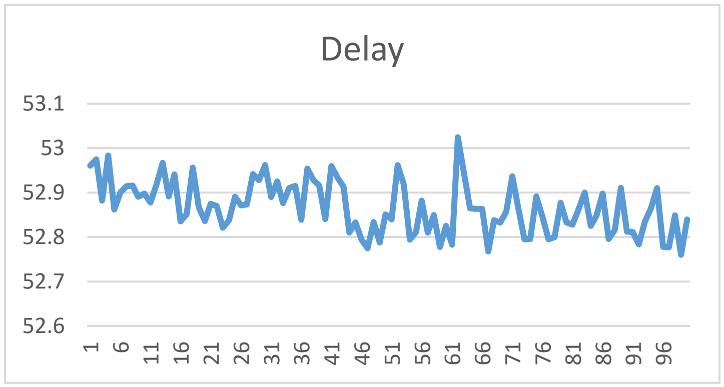


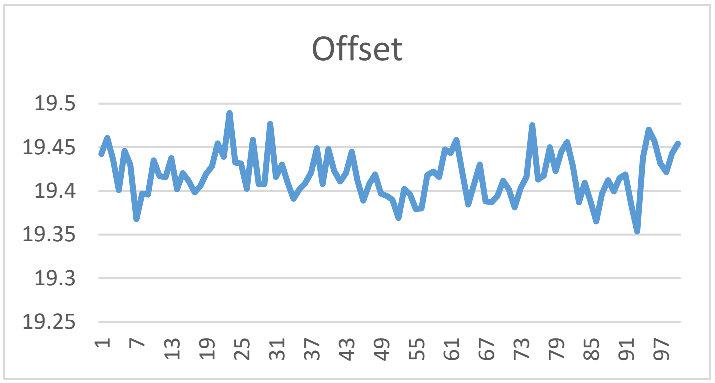
Write latency:



The mean offset observed for this topology was 19.42 s. However, like with the second topology, the initial offset observed was different and was -3.82 s.

Clock skew:





Conclusion

The read and write latencies (with the exception of the second topology's write) seemed to increase as the topologies expanded. As expected, the clock skew tended to increase as the topologies got bigger and more spread out, however it was observed that giving the cluster time seemed to cause a higher offset, possibly owing to clock drift caused by an inadequate NTP setup.

Clock skew could be affected by different routing between the nodes or clock drift – often there is asymmetry in routes linking servers in different regions, such that a message may not return the same way it was sent in, which violates the assumption made in calculating these values that send and receive times are supposed to be the same. When messages have to be sent across large distances or through oceanic links, this can increase latencies greatly. This explains why with expanded topologies larger latency values are observed.