Measuring Cassandra Cluster

Rohit Mehra

University of Colorado, Boulder

Abstract

Apache Cassandra is a highly scalable, distributed, high performance database that falls under the NoSQL category. It is designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.Due to replication and clustering feature, it is highly scalable and its performance increases linearly as we add the nodes.

The aim of the project is to get familiar with Docker and on using NTP and setting Cassandra cluster. The project focuses on getting measurements in terms of read-write latencies and ping latencies across Cassandra Clusters setup in multiple topologies and networks. We have implemented 4 topologies which is elaborated in the sections below.

Keywords:

Cassandra, NoSQL, Docker, VirtualMachine

1. The First Topology

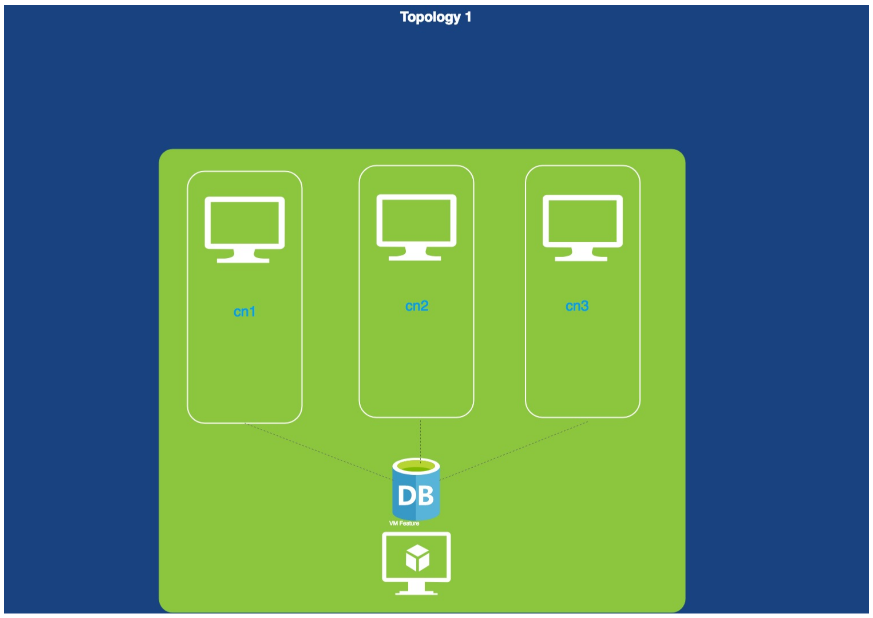


Figure 1: Topology 1

In the topology, we had a single host system running three different docker containers. Each container had a Cassandra node in it and was called cn1, cn2 and cn3.Also, we booted up 1 machine which was used as server to measure clock skew. The steps we followed were as follows.

1. Dell XPS was used as host system with 16GB RAM

2. The next step was to spring up the Docker instances. The instructions for which could be found in the docker documentation.
3. The third step was to install NTP and Cassandra.
4. The last step was to establish a cluster using all the client nodes.
5. Cassandra stress command was used to measure read, write and mix latencies among a cluster node.
6. One client and server combination was used to test clock skew.

The figure describing the topology is $Figure1$.

1.1. Testing Topology 1

1. All the docker containers are assigned IP from the docker subnet.
2. We used public NTP server IP in the server node so that server can be in sync with a public server.
3. NTP.conf file of all the clients points to the server IP so that it can be in sync.
4. ntpstat can be used to check the synchronization state of the client and the server.
5. The reason we needed to use NTP to configure the clock skew is as follows: Computer's use Cesium 133 for clock synchronization.

   As we go down the tree as shown in $Figure2$, we start to get skew between the root node and the nodes along the tree. To ensure that all the VMs/Containers follow the same time, we use NTP for Synchronization.

   The NTP conf can be configured at \etc\ntp.conf. Once the conf files are set, the servers are rebooted.The sync status can be seen using the *ntpstat* command.

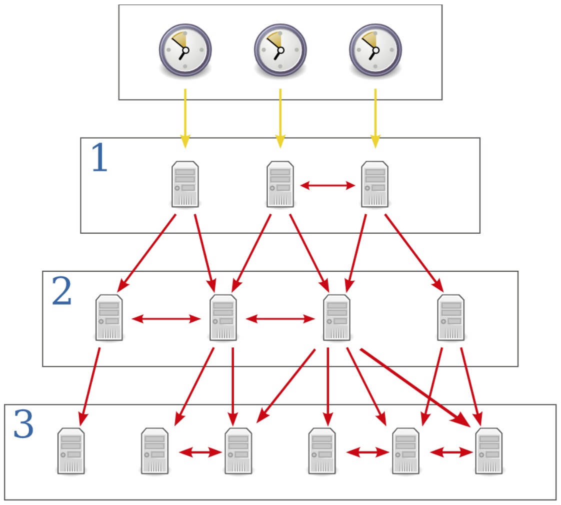
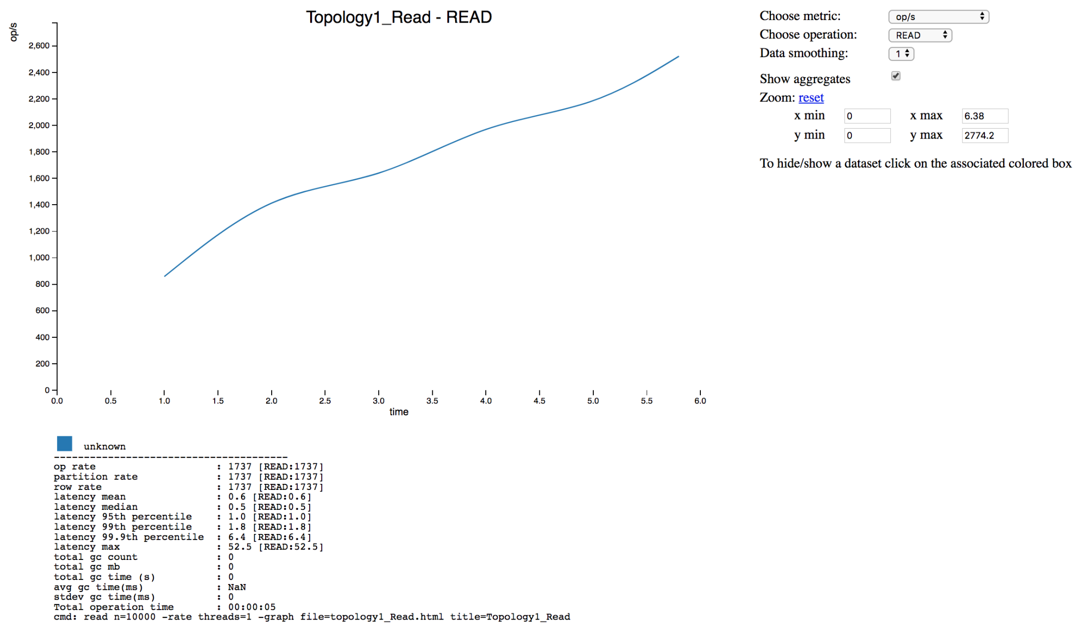


Figure 2: NTP Hierarchy

6. Now we wanted to check if the key spaces are shared between the Cassandra ring.
7. We used the $cassandra - stress$ tool to get the read, write and mixed latencies of the same.
8. As the number of iterations increased, the read and write latencies started to gradually increase as well.

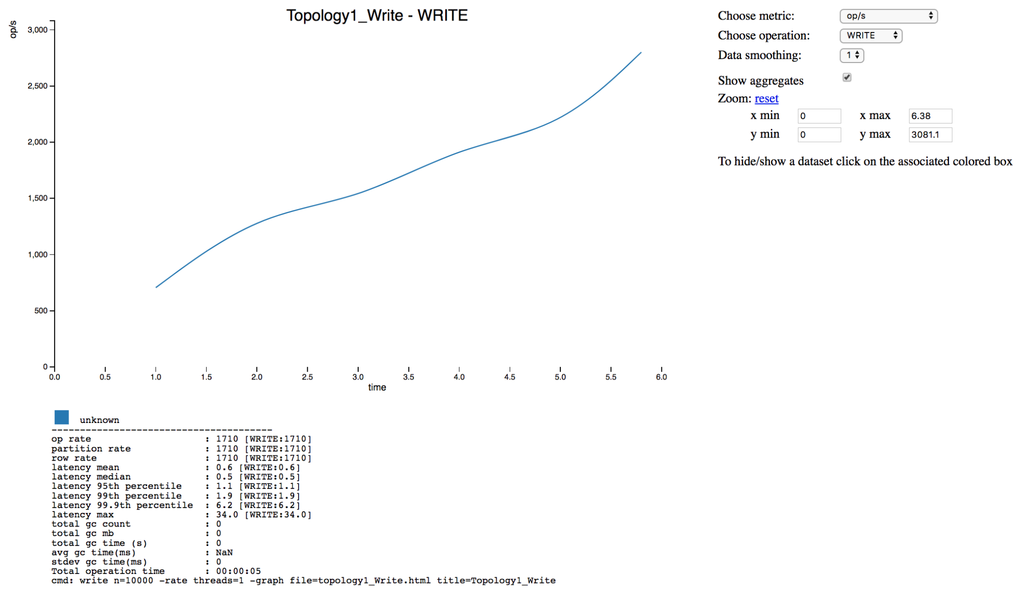
9. This can be seen in $Figure3$, $Figure4$, $Figure5$



Choose metric: op/s
Choose operation: READ
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 6.38
y min 0 y max 2774.2

```
🟦 unknown
----------------------------------------
op rate                  : 1737 [READ:1737]
partition rate           : 1737 [READ:1737]
row rate                 : 1737 [READ:1737]
latency mean             : 0.6 [READ:0.6]
latency median           : 0.5 [READ:0.5]
latency 95th percentile  : 1.0 [READ:1.0]
latency 99th percentile  : 1.8 [READ:1.8]
latency 99.9th percentile : 6.4 [READ:6.4]
latency max              : 52.5 [READ:52.5]
total gc count           : 0
total gc mb              : 0
total gc time (s)        : 0
avg gc time(ms)          : NaN
stdev gc time(ms)        : 0
Total operation time     : 00:00:05
cmd: read n=10000 -rate threads=1 -graph file=topology1_Read.html title=Topology1_Read
```

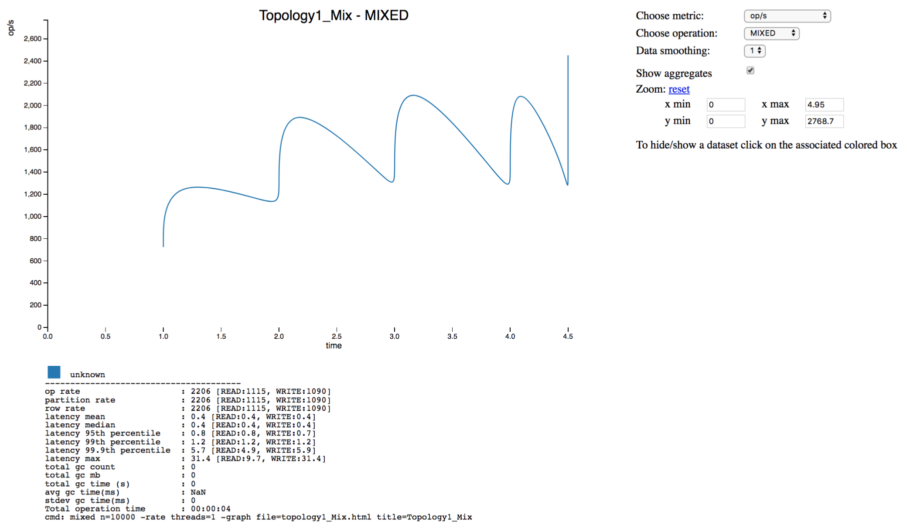
Figure 3: Cassandra Stress Topology 1 Read



Choose metric: op/s
Choose operation: WRITE
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 6.38
y min 0 y max 3081.1

```
🟦 unknown
----------------------------------------
op rate                  : 1710 [WRITE:1710]
partition rate           : 1710 [WRITE:1710]
row rate                 : 1710 [WRITE:1710]
latency mean             : 0.6 [WRITE:0.6]
latency median           : 0.5 [WRITE:0.5]
latency 95th percentile  : 1.1 [WRITE:1.1]
latency 99th percentile  : 1.9 [WRITE:1.9]
latency 99.9th percentile : 6.2 [WRITE:6.2]
latency max              : 34.0 [WRITE:34.0]
total gc count           : 0
total gc mb              : 0
total gc time (s)        : 0
avg gc time(ms)          : NaN
stdev gc time(ms)        : 0
Total operation time     : 00:00:05
cmd: write n=10000 -rate threads=1 -graph file=topology1_Write.html title=Topology1_Write
```

Figure 4: Cassandra Stress Topology 1 Write



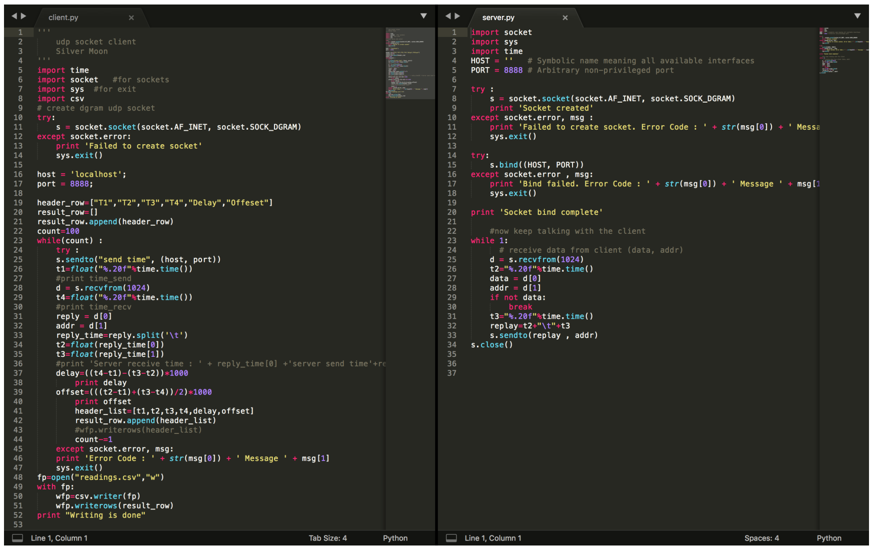
Choose metric: op/s
Choose operation: MIXED
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 4.95
y min 0 y max 2768.7

To hide/show a dataset click on the associated colored box

```
■ unknown
----------------------------------------
op rate                    : 2206 [READ:1115, WRITE:1090]
partition rate             : 2206 [READ:1115, WRITE:1090]
row rate                   : 2206 [READ:1115, WRITE:1090]
latency mean               : 0.4 [READ:0.4, WRITE:0.4]
latency median             : 0.4 [READ:0.4, WRITE:0.4]
latency 95th percentile    : 0.8 [READ:0.8, WRITE:0.7]
latency 99th percentile    : 1.2 [READ:1.2, WRITE:1.2]
latency 99.9th percentile  : 5.7 [READ:4.9, WRITE:5.9]
latency max                : 31.4 [READ:9.7, WRITE:31.4]
total gc count             : 0
total gc mb                : 0
total gc time (s)          : 0
avg gc time(ms)            : NaN
stdev gc time(ms)          : 0
Total operation time       : 00:00:04
cmd: mixed n=10000 -rate threads=1 -graph file=topology1_Mix.html title=Topology1_Mix
```

Figure 5: Cassandra Stress Topology 1 Mixed

10. The increase in all the measurements is linear as all the containers are on the host system so there is not much latency.
11. The last step was to measure the clock skew. This was done using the client-server code described in $Figure 4$



client.py

```python
'''
    udp socket client
    Silver Moon
'''
import time
import socket   #for sockets
import sys  #for exit
import csv
# create dgram udp socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error:
    print 'Failed to create socket'
    sys.exit()

host = 'localhost';
port = 8888;

header_row=["T1","T2","T3","T4","Delay","Offeset"]
result_row=[]
result_row.append(header_row)
count=100
while(count) :
    try :
    s.sendto("send time", (host, port))
    t1=float("%.20f"%time.time())
    #print time_send
    d = s.recvfrom(1024)
    t4=float("%.20f"%time.time())
    #print time_recv
    reply = d[0]
    addr = d[1]
    reply_time=reply.split('\t')
    t2=float(reply_time[0])
    t3=float(reply_time[1])
    #print 'Server receive time : ' + reply_time[0] +'server send time'+re
    delay=((t4-t1)-(t3-t2))*1000
        print delay
    offset=(((t2-t1)+(t3-t4))/2)*1000
        print offset
        header_list=[t1,t2,t3,t4,delay,offset]
        result_row.append(header_list)
        #wfp.writerows(header_list)
        count-=1
    except socket.error, msg:
    print 'Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()
fp=open("readings.csv","w")
with fp:
    wfp=csv.writer(fp)
    wfp.writerows(result_row)
print "Writing is done"
```

Line 1, Column 1 Tab Size: 4 Python

server.py

```python
import socket
import sys
import time
HOST = ''   # Symbolic name meaning all available interfaces
PORT = 8888 # Arbitrary non-privileged port

try :
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    print 'Socket created'
except socket.error, msg :
    print 'Failed to create socket. Error Code : ' + str(msg[0]) + ' Messa
    sys.exit()

try:
    s.bind((HOST, PORT))
except socket.error , msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1
    sys.exit()

print 'Socket bind complete'

    #now keep talking with the client
while 1:
      # receive data from client (data, addr)
    d = s.recvfrom(1024)
    t2="%.20f"%time.time()
    data = d[0]
    addr = d[1]
    if not data:
        break
    t3="%.20f"%time.time()
    replay=t2+"\t"+t3
    s.sendto(replay , addr)
s.close()
```

Line 1, Column 1 Spaces: 4 Python

Figure 6: Client-Server code

12. The results of the same can be found in $Figure 5$

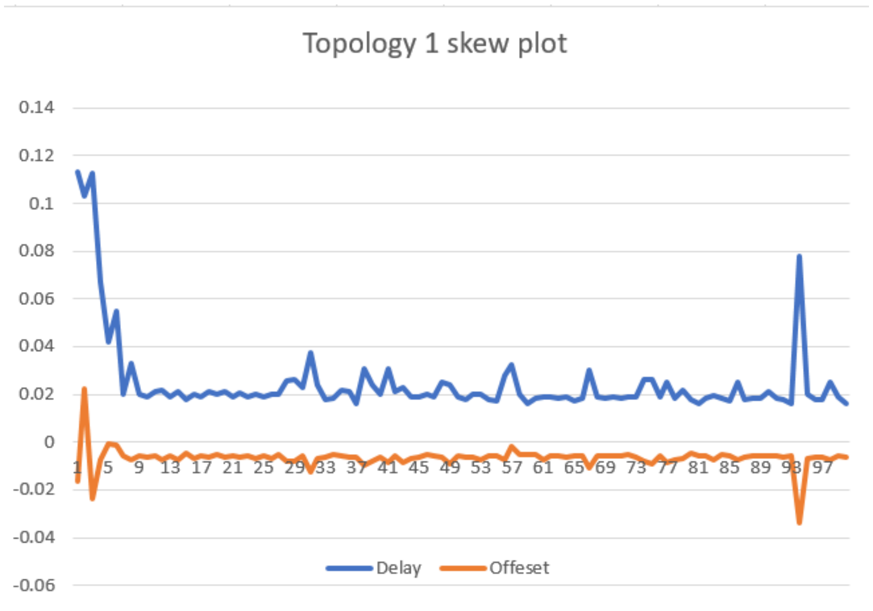


Figure 7: Skew output

13. Offset and Delay is almost constant apart from the start and the end.

2. The Second Topology

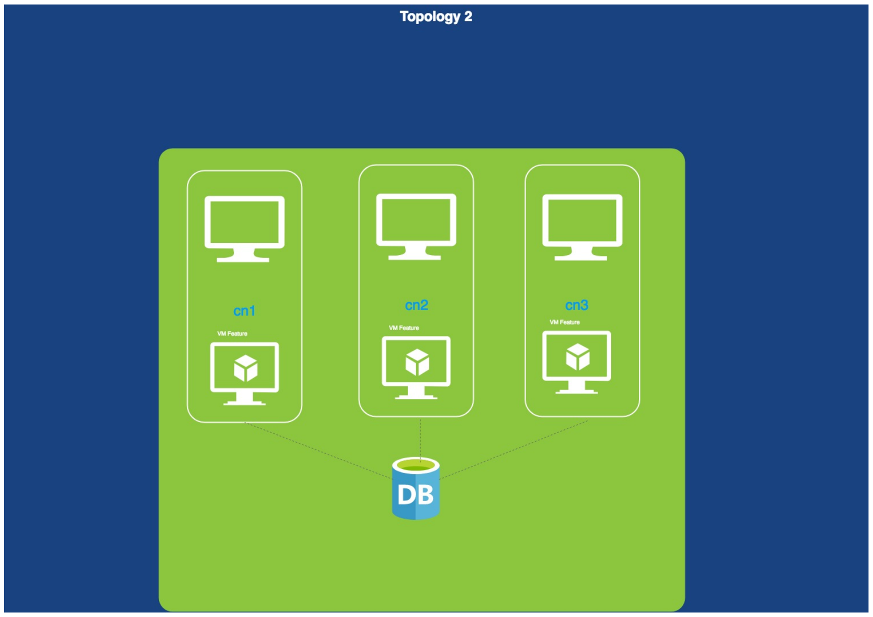


Figure 8: Second Topology

In the topology, we had a three VMs running three different docker containers. Each container had a Cassandra node in it and was called cn1, cn2 and cn3. The VM was sprung up using Oracle's VirtualBox technology.
The steps we followed were as follows.

1. Setup three Virtual machine using Oracle's Virtual Box. The Operating System used was Ubuntu 16.04, with 20GB disk space and 4GB RAM.
2. NAT Network was used as network setting with different MAC address for all the VM. This was done to assign different IP to each VM.

3. The next step was to install Cassandra, NTP on all the VM's.
4. The last step was to establish connections between the containers containing Cassandra to form a ring

The figure describing the topology is $Figure8$.

2.1. Testing Topology 2

1. The first step was to ensure that the docker container can ping one another. This was done using the *ping* command, using the Docker's IP that can be obtained using the $ifconfig$ command. Once the ping is received, we can go ahead with the next steps.
2. We first installed NTP on each of the docker containers in the each VM. Next we treated one of the containers as the NTP server and the rest as the NTP clients.
3. To ensure that all the VMs/Containers follow the same time, we use NTP for Synchronization.
4. Like in the previous topology ntpstat was used to check if clocks are in sync.

   The NTP conf can be configured at \etc\ntp.conf. Once the conf files are set, the servers are rebooted.The sync status can be seen using the *ntpstat* command.
5. Now we wanted to check if the key spaces are shared between the Cassandra ring.
6. We used the $cassandra - stress$ tool to get the read, write and mixed latencies of the same.
7. As the number of iterations increased, the read and write latencies started to gradually increase as well.
8. This can be seen in $Figure10$, $Figure11$ and $Figure12$
9. The last step was to measure the clock skew. This was done using the client-server code described in $Figure4$
10. The results of the same can be found in $Figure9$
11. As different VM's are used, our operations per second for read, write and mix latencies increase.
12. Offset and Delay increase variably as the network delay increases.

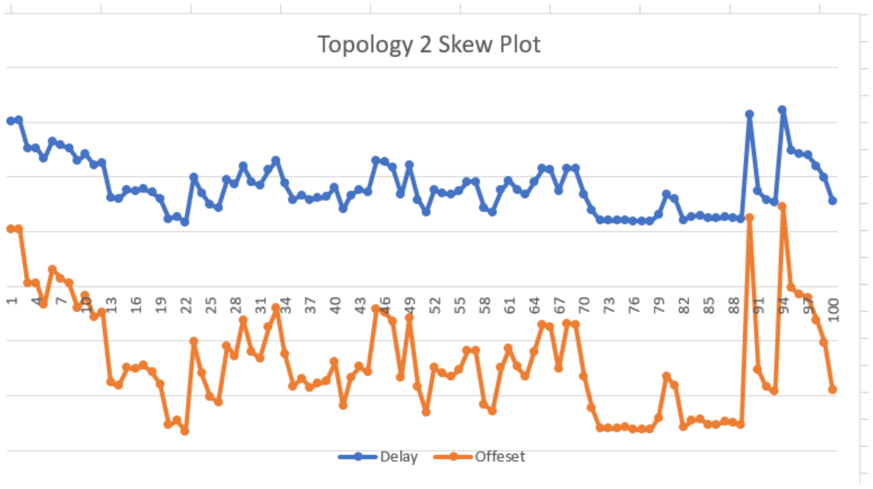
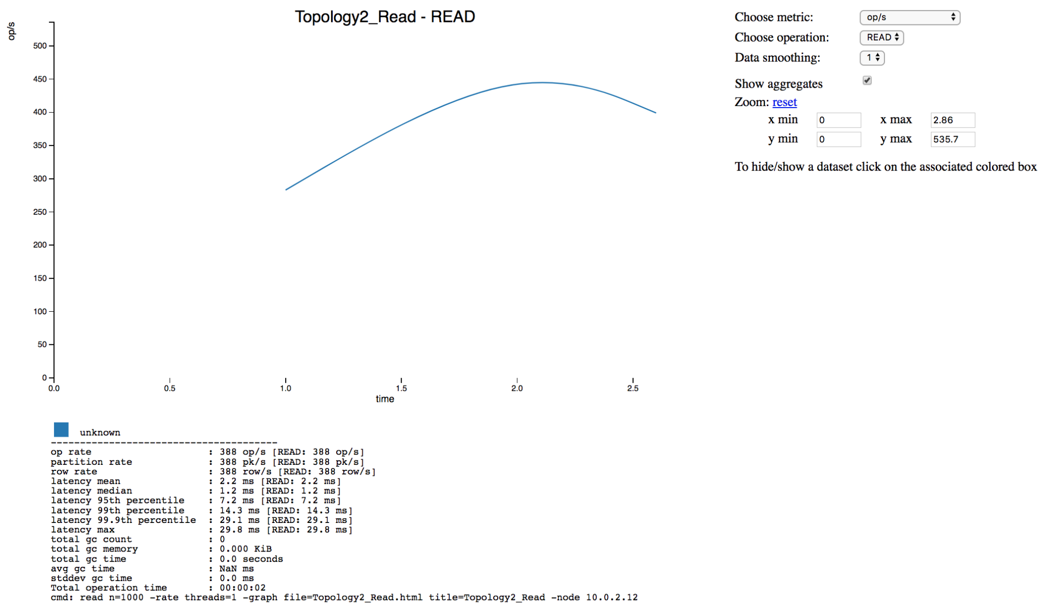


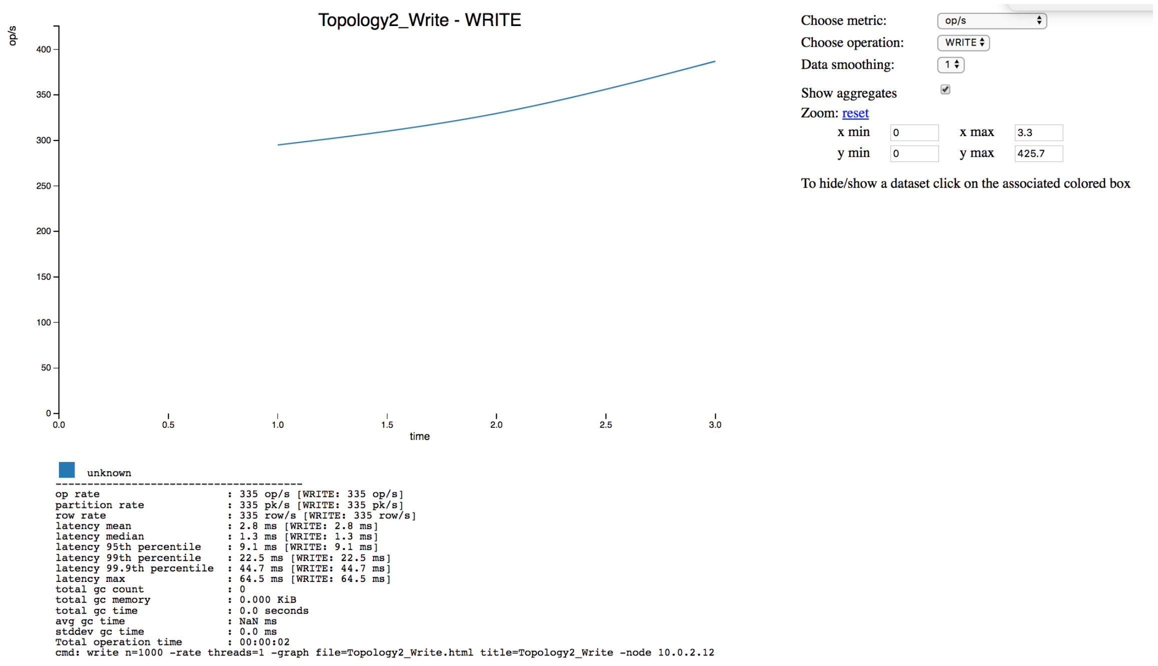
Figure 9: Skew output



Choose metric: op/s
Choose operation: READ
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 2.86
y min 0 y max 535.7
To hide/show a dataset click on the associated colored box

```
🟦 unknown
----------------------------------------
op rate                   : 388 op/s [READ: 388 op/s]
partition rate            : 388 pk/s [READ: 388 pk/s]
row rate                  : 388 row/s [READ: 388 row/s]
latency mean              : 2.2 ms [READ: 2.2 ms]
latency median            : 1.2 ms [READ: 1.2 ms]
latency 95th percentile   : 7.2 ms [READ: 7.2 ms]
latency 99th percentile   : 14.3 ms [READ: 14.3 ms]
latency 99.9th percentile : 29.1 ms [READ: 29.1 ms]
latency max               : 29.8 ms [READ: 29.8 ms]
total gc count            : 0
total gc memory           : 0.000 KiB
total gc time             : 0.0 seconds
avg gc time               : NaN ms
stddev gc time            : 0.0 ms
Total operation time      : 00:00:02
cmd: read n=1000 -rate threads=1 -graph file=Topology2_Read.html title=Topology2_Read -node 10.0.2.12
```

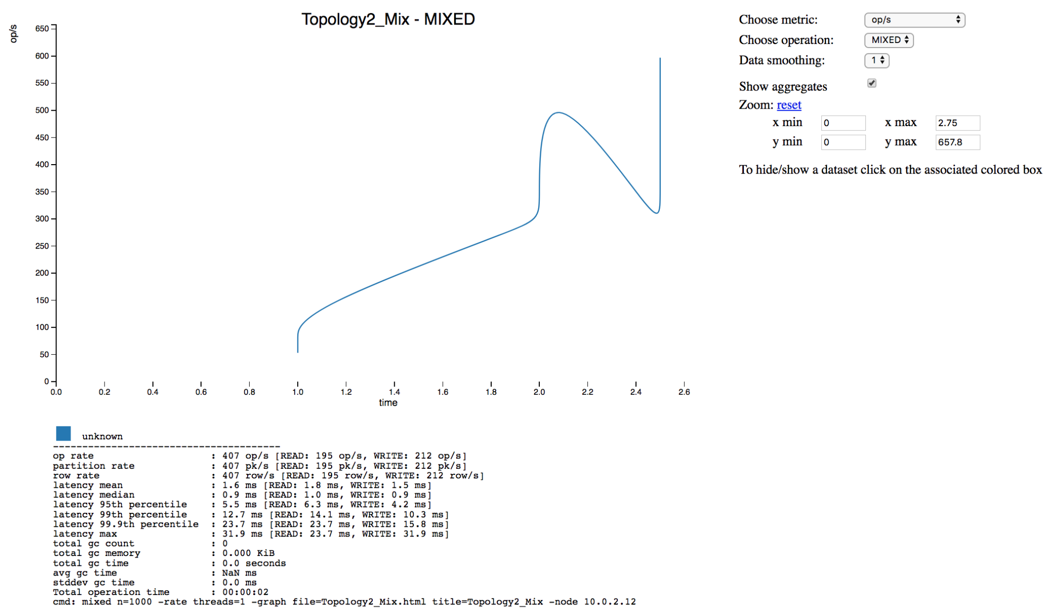
Figure 10: Cassandra Stress Topology 2 Read



Choose metric: op/s
Choose operation: WRITE
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 3.3
y min 0 y max 425.7
To hide/show a dataset click on the associated colored box

```
🟦 unknown
----------------------------------------
op rate                   : 335 op/s [WRITE: 335 op/s]
partition rate            : 335 pk/s [WRITE: 335 pk/s]
row rate                  : 335 row/s [WRITE: 335 row/s]
latency mean              : 2.8 ms [WRITE: 2.8 ms]
latency median            : 1.3 ms [WRITE: 1.3 ms]
latency 95th percentile   : 9.1 ms [WRITE: 9.1 ms]
latency 99th percentile   : 22.5 ms [WRITE: 22.5 ms]
latency 99.9th percentile : 44.7 ms [WRITE: 44.7 ms]
latency max               : 64.5 ms [WRITE: 64.5 ms]
total gc count            : 0
total gc memory           : 0.000 KiB
total gc time             : 0.0 seconds
avg gc time               : NaN ms
stddev gc time            : 0.0 ms
Total operation time      : 00:00:02
cmd: write n=1000 -rate threads=1 -graph file=Topology2_Write.html title=Topology2_Write -node 10.0.2.12
```

Figure 11: Cassandra Stress Topology 2 Write



Choose metric: op/s
Choose operation: MIXED
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 2.75
y min 0 y max 657.8

```
🟦  unknown
---------------------------------------
op rate                   : 407 op/s [READ: 195 op/s, WRITE: 212 op/s]
partition rate            : 407 pk/s [READ: 195 pk/s, WRITE: 212 pk/s]
row rate                  : 407 row/s [READ: 195 row/s, WRITE: 212 row/s]
latency mean              : 1.6 ms [READ: 1.8 ms, WRITE: 1.5 ms]
latency median            : 0.9 ms [READ: 1.0 ms, WRITE: 0.9 ms]
latency 95th percentile   : 5.5 ms [READ: 6.3 ms, WRITE: 4.2 ms]
latency 99th percentile   : 12.7 ms [READ: 14.1 ms, WRITE: 10.3 ms]
latency 99.9th percentile : 23.7 ms [READ: 23.7 ms, WRITE: 15.8 ms]
latency max               : 31.9 ms [READ: 23.7 ms, WRITE: 31.9 ms]
total gc count            : 0
total gc memory           : 0.000 KiB
total gc time             : 0.0 seconds
avg gc time               : NaN ms
stddev gc time            : 0.0 ms
Total operation time      : 00:00:02
cmd: mixed n=1000 -rate threads=1 -graph file=Topology2_Mix.html title=Topology2_Mix -node 10.0.2.12
```

Figure 12: Cassandra Stress Topology 2 Mixed

3. The Third Topology

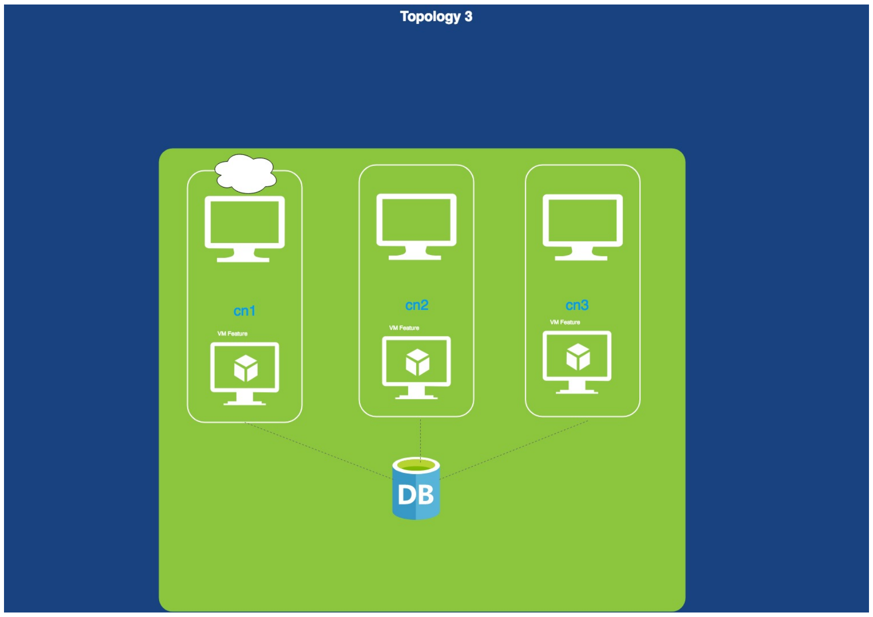


Figure 13: Toplogy 3

In the topology, we had a three VMs running three different docker containers in three different datacenters in Clemson, Utah and Wisconsin. Each container had a Cassandra node in it and was called cn1, cn2 and cn3. The VM was sprung up using CloudLab.
The steps we followed were as follows.

1. Setup a Virtual machines using Cloudlab. The Operating System used was Ubuntu 14.04.
2. The second step was to install Cassandra in each of the machines.
3. The last step was to establish connections between the cloud instances containing Cassandra to form a ring.
4. We had to open ports on the system for communication.

The figure describing the topology is $Figure 13$.

3.1. Testing Topology 3

1. The first step was to ensure that all machines can ping one another. This was done using the $ping$ command, using the machine's Public IP that can be obtained using the $ifconfig$ command. Once the ping is received, we can go ahead with the next steps.
2. To calculate time skew, we made one of the machine as server and modified NTP.conf file to point to NTP public IP.
3. We had to remove NTP from DHCP so that changes in NTP file can persist.
4. NTP.conf file of all the clients was edited to point to IP of the server so that they can sync their time with server.
5. We used the $cassandra - stress$ tool to get the read, write and mixed latencies of the same.
6. As the number of iterations increased, the read and write latencies started to gradually increase as well.The main reason was the machines were in different physical locations.
7. This can be seen in $Figure 15$, $Figure 16$ and $Figure 17$

8. The last step was to measure the clock skew. This was done using the client-server code described in $Fig4$
9. The results of the same can be found in $Figure14$
10. Delay and offset also increase due to change in the machines geography.

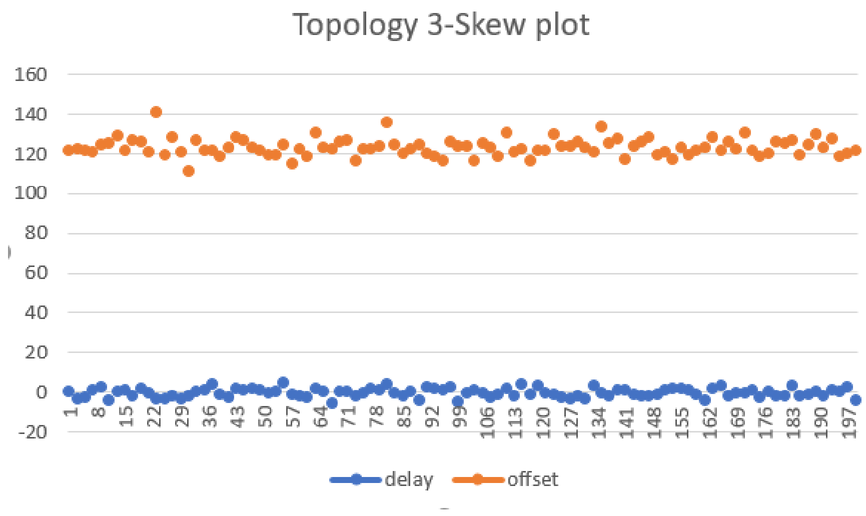


Figure 14: Skew output

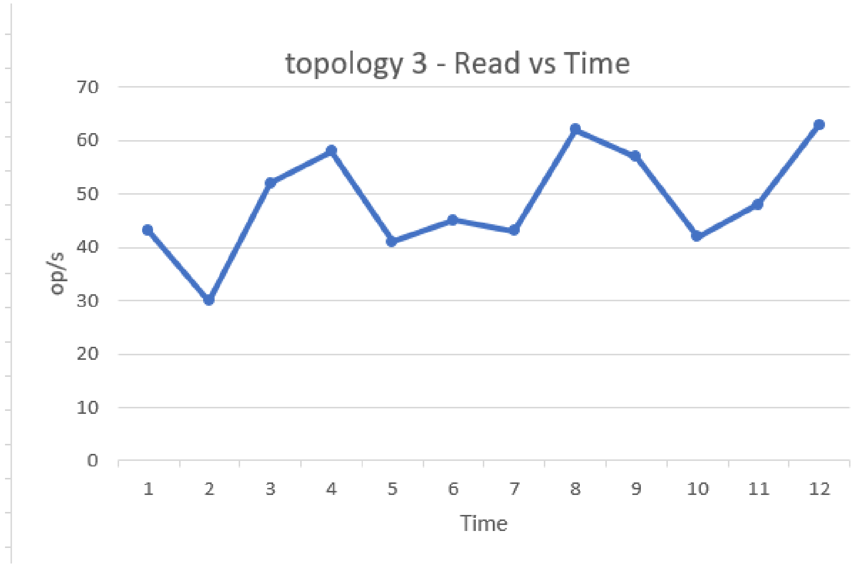


Figure 15: Cassandra Stress Topology 3 Read

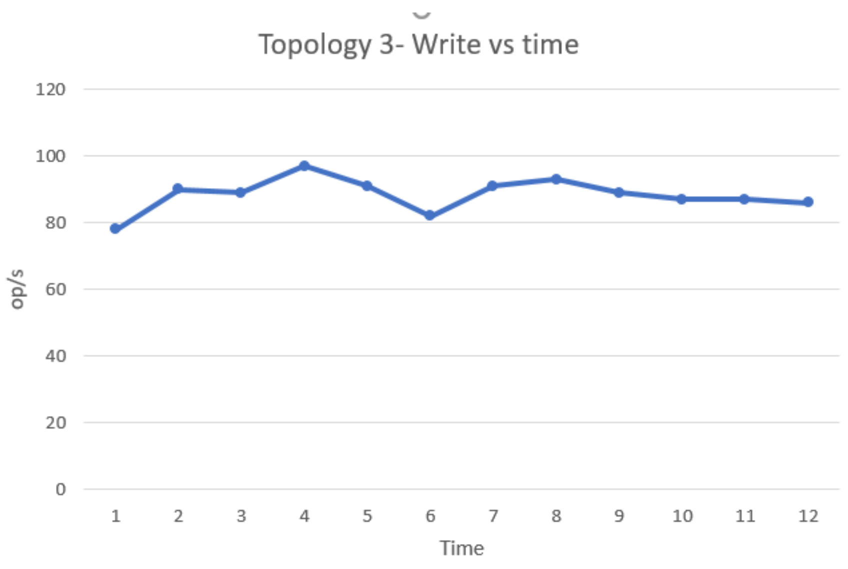


Figure 16: Cassandra Stress Topology 3 Write

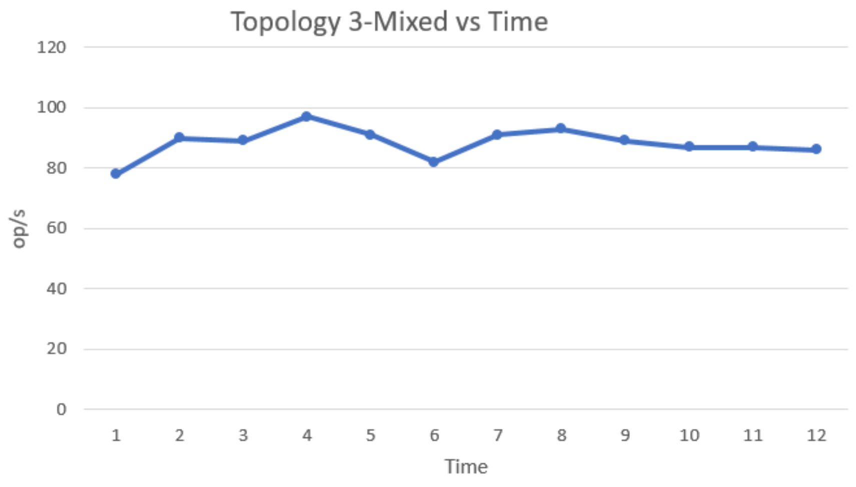


Figure 17: Cassandra Stress Topology 3 Mixed

4. The Fourth Topology

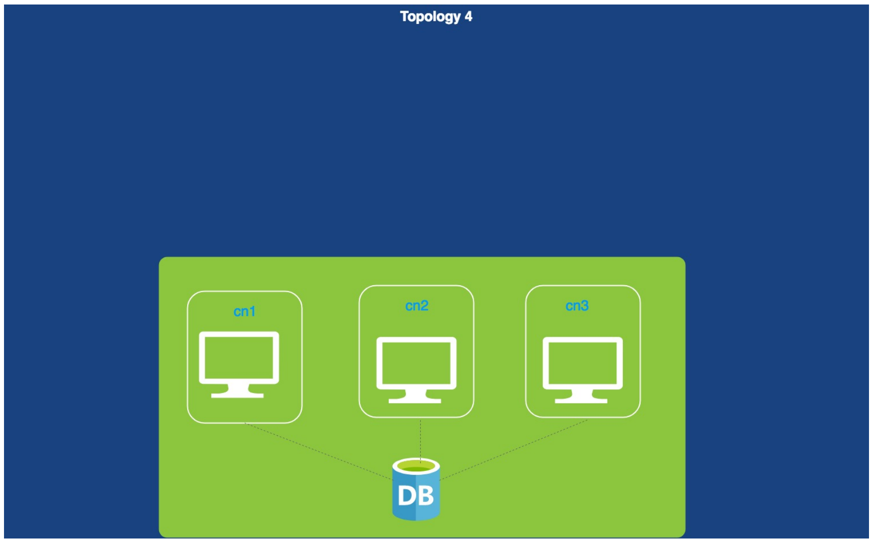


Figure 18: Topology 4

In the topology, we had a three physical machines running in the same network. Each machine had a Cassandra node in it and was called cn1, cn2 and cn3. The VM was sprung up using Oracle's Virtualbox.
The steps we followed were as follows.

1. Install cassandra and ntp in all the machines.
2. Open ports on all the machines for communication.

The figure describing the topology is $Figure18$.

4.1. Testing Topology 4

1. The first step was to ensure that the machines can ping one another. This was done using the *ping* command, using the machine's IP that can be obtained using the $ifconfig$ command. Once the ping is received, we can go ahead with the next steps.
2. We used 1 machine as the server and synced its time with the public NTP server.
3. We modified the NTP.conf file of all the client machines to point to server IP.The sync status can be seen using the *ntpstat* command.
4. Now we wanted to check if the key spaces are shared between the Cassandra ring.
5. We used the $cassandra - stress$ tool to get the read, write and mixed latencies of the same.
6. As the number of iterations increased, the read and write latencies linearly increased.
7. It was similar to the first topology as machines were in same LAN.
8. This can be seen in $Figure19$, $Figure20$ and $Figure21$
9. The last step was to measure the clock skew. This was done using the client-server code described in $Figure4$
10. The results of the same can be found in $Figure22$

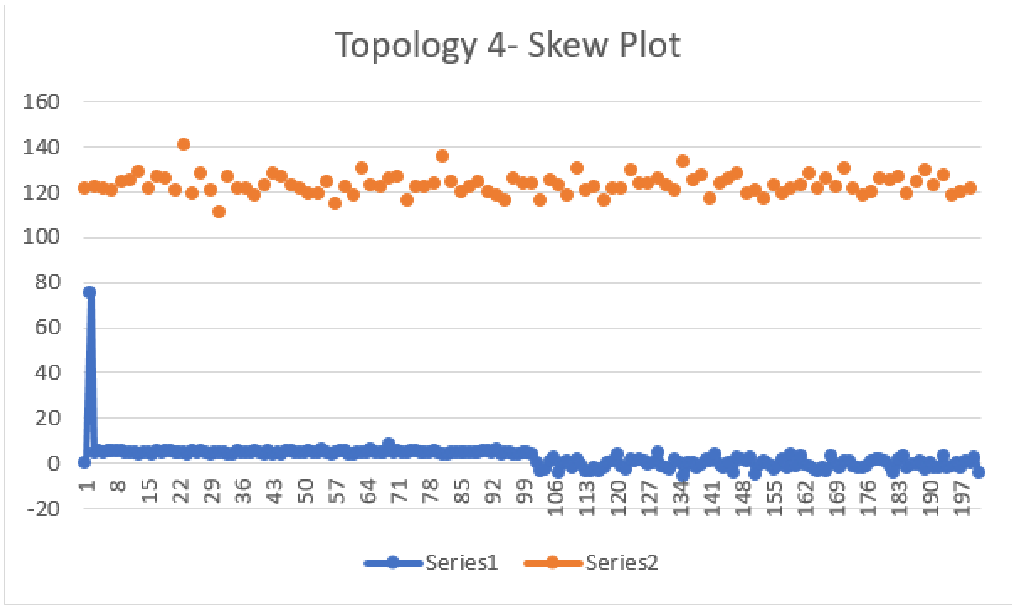
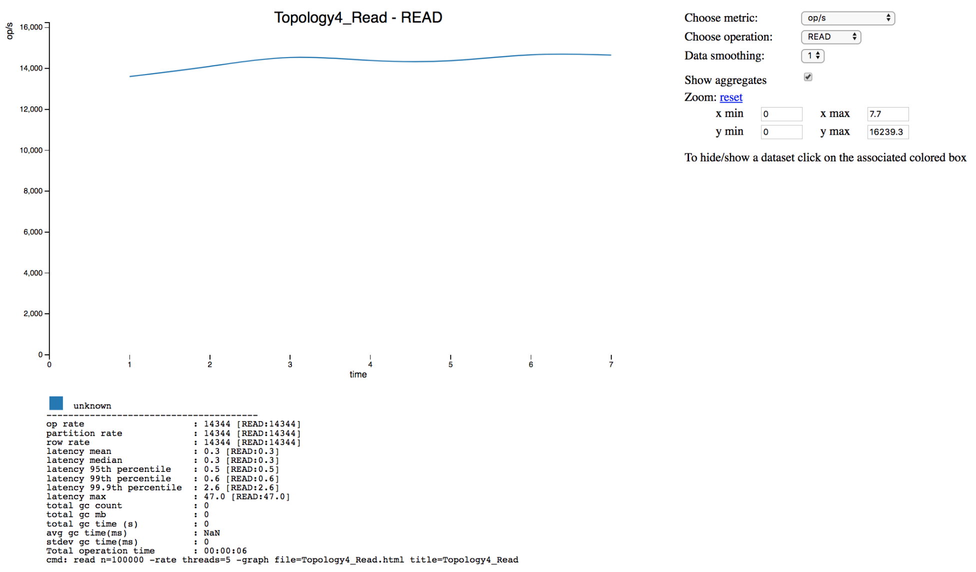


Figure 19: Skew output

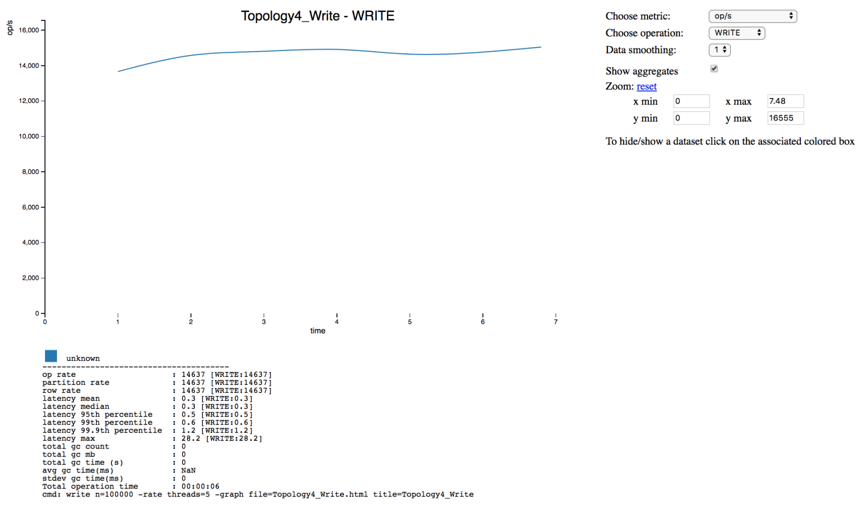


Choose metric: op/s
Choose operation: READ
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 7.7
y min 0 y max 16239.3

To hide/show a dataset click on the associated colored box

```
■ unknown
---------------------------------------
op rate                   : 14344 [READ:14344]
partition rate            : 14344 [READ:14344]
row rate                  : 14344 [READ:14344]
latency mean              : 0.3 [READ:0.3]
latency median            : 0.3 [READ:0.3]
latency 95th percentile   : 0.5 [READ:0.5]
latency 99th percentile   : 0.6 [READ:0.6]
latency 99.9th percentile : 2.6 [READ:2.6]
latency max               : 47.0 [READ:47.0]
total gc count            : 0
total gc mb               : 0
total gc time (s)         : 0
avg gc time(ms)           : NaN
stdev gc time(ms)         : 0
Total operation time      : 00:00:06
cmd: read n=100000 -rate threads=5 -graph file=Topology4_Read.html title=Topology4_Read
```

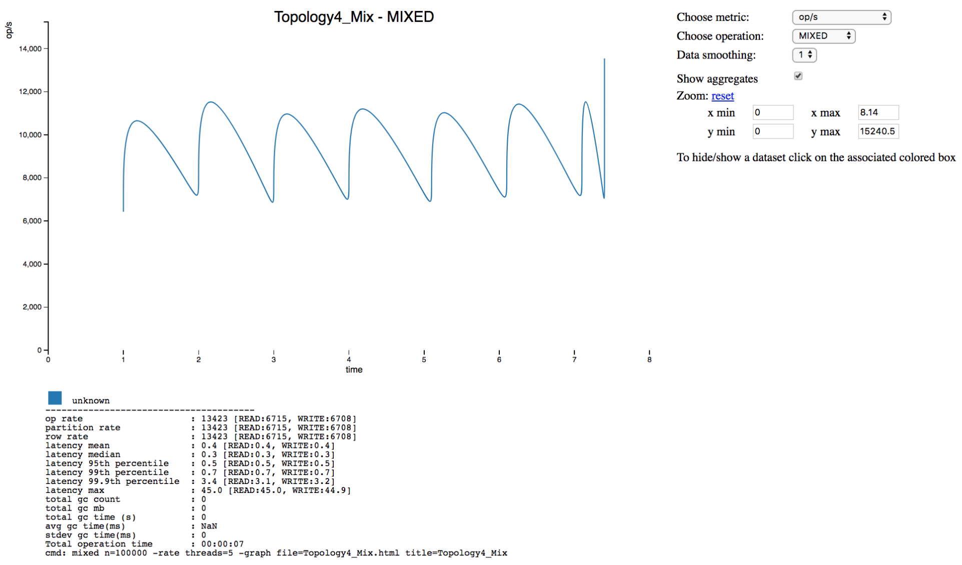
Figure 20: Cassandra Stress Topology 4 Read



Choose metric: op/s
Choose operation: WRITE
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 7.48
y min 0 y max 16555

```
🟦 unknown
--------------------------------------
op rate                      : 14637 [WRITE:14637]
partition rate               : 14637 [WRITE:14637]
row rate                     : 14637 [WRITE:14637]
latency mean                 : 0.3 [WRITE:0.3]
latency median               : 0.3 [WRITE:0.3]
latency 95th percentile      : 0.5 [WRITE:0.5]
latency 99th percentile      : 0.6 [WRITE:0.6]
latency 99.9th percentile    : 1.2 [WRITE:1.2]
latency max                  : 28.2 [WRITE:28.2]
total gc count               : 0
total gc mb                  : 0
total gc time (s)            : 0
avg gc time(ms)              : NaN
stdev gc time(ms)            : 0
Total operation time         : 00:00:06
cmd: write n=100000 -rate threads=5 -graph file=Topology4_Write.html title=Topology4_Write
```

Figure 21: Cassandra Stress Topology 4 Write



Choose metric: op/s
Choose operation: MIXED
Data smoothing: 1
Show aggregates ✓
Zoom: reset
x min 0 x max 8.14
y min 0 y max 15240.5

```
🟦 unknown
--------------------------------------
op rate                      : 13423 [READ:6715, WRITE:6708]
partition rate               : 13423 [READ:6715, WRITE:6708]
row rate                     : 13423 [READ:6715, WRITE:6708]
latency mean                 : 0.4 [READ:0.4, WRITE:0.4]
latency median               : 0.3 [READ:0.3, WRITE:0.3]
latency 95th percentile      : 0.5 [READ:0.5, WRITE:0.5]
latency 99th percentile      : 0.7 [READ:0.7, WRITE:0.7]
latency 99.9th percentile    : 3.4 [READ:3.1, WRITE:3.2]
latency max                  : 45.0 [READ:45.0, WRITE:44.9]
total gc count               : 0
total gc mb                  : 0
total gc time (s)            : 0
avg gc time(ms)              : NaN
stdev gc time(ms)            : 0
Total operation time         : 00:00:07
cmd: mixed n=100000 -rate threads=5 -graph file=Topology4_Mix.html title=Topology4_Mix
```

Figure 22: Cassandra Stress Topology 4 Mixed

5. Conclusion

For the simple topology, read, write and mix latency will be less as compared to the complex topologies. Also delay and offset will be very less for the simple topology and will increase as the topology becomes more complex.We can see in topology 1 and topology 4, either all the containers are in the same system or are in the same LAN on different machines.So the latency, delay and offset is very less for these topologies. For topology 3, the latencies increase and the number of operation decrease as the machines are in Virtual Machines. For topology 4, the latencies increase and number of operations decrease as the machines are in different physical locations. We tested delay and offset after running the stress test so that the machines have some data.