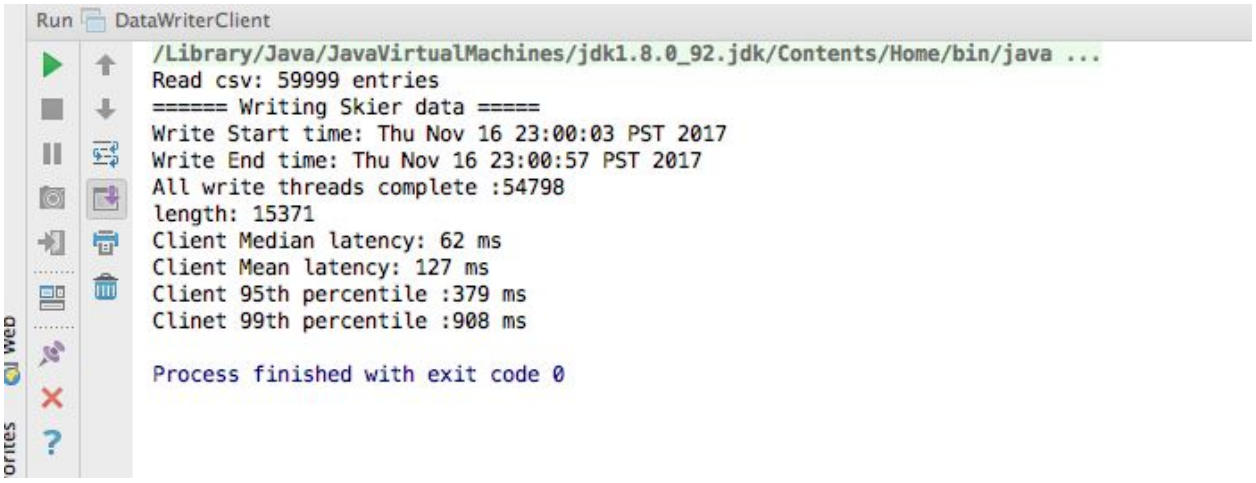
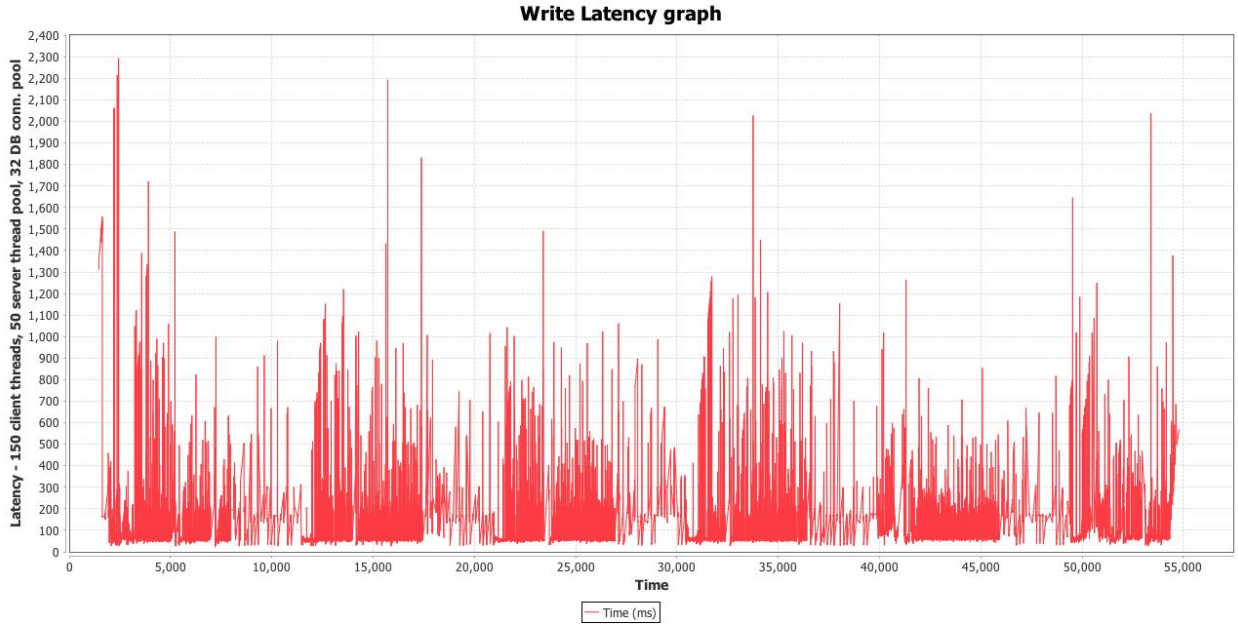
Assignment 3 : Starting to Scale the Server Amala Joshy

Step 1 - Scale the Server

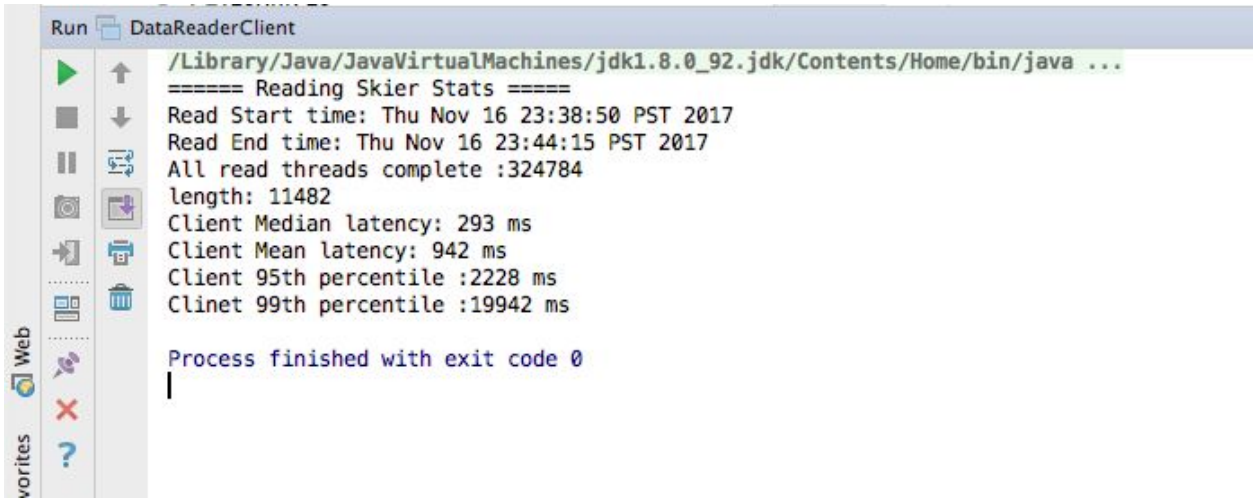
So the first step was to scale the server, by configuring 3 instances with elastic load balancer.
Client side with ELB and 3 server instances, Run screenshot for write Day999 :



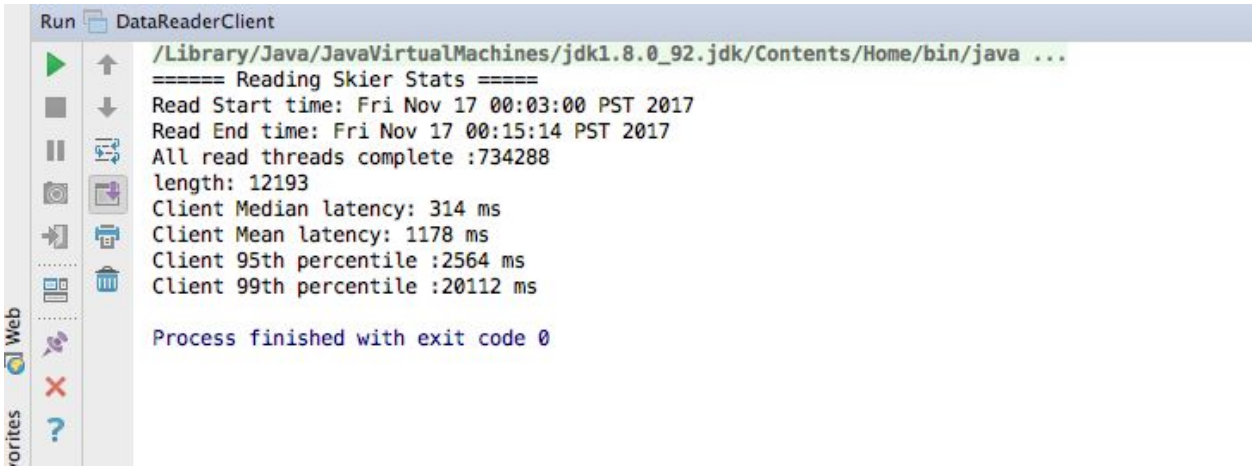


To compare the performance, Mean latency to write the data seems to be way much less than than before configuring the load balancer, mean latency used to be around ~500ms where as now its 127ms. The performance have gone up, you can see the rest of the parameter values from the run screen shot.

Sample output from few reads on Client with ELB configured : Day 999



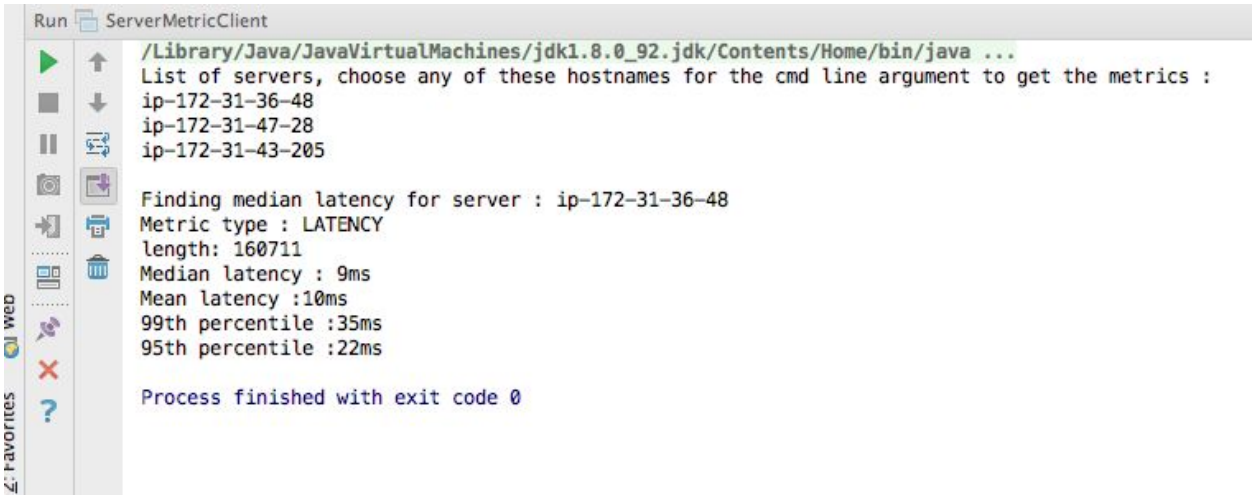
Day 1 :



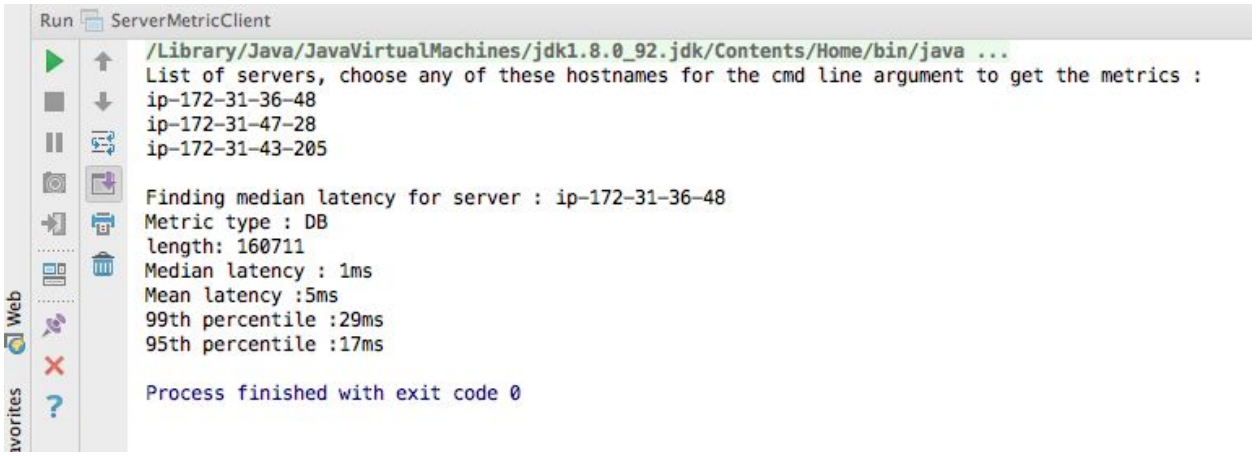
Step 2 - Instrument the Server

The command line program ServerMetricClient lets you give the desired server metric specification to get the performance parameter values like Latency for Read and Writes, DB latencies and Error count. Metric data is written to an RDS metric table from where the parameters will be calculated. Monitoring code is the MetricReporter class which captures the latency points without much overhead and error counts to db. Enum Metric - > helps measure the performance parameters easily. Metric reporter writes the collected latency values to the database in fixed intervals.

Latency metric for Read day 999 :

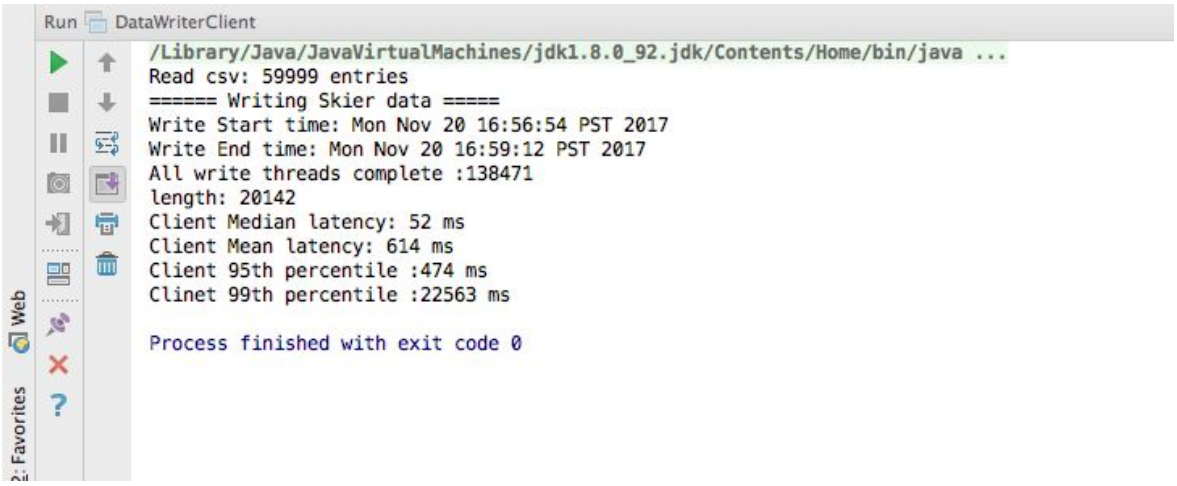


DB latency metric for Read day 999

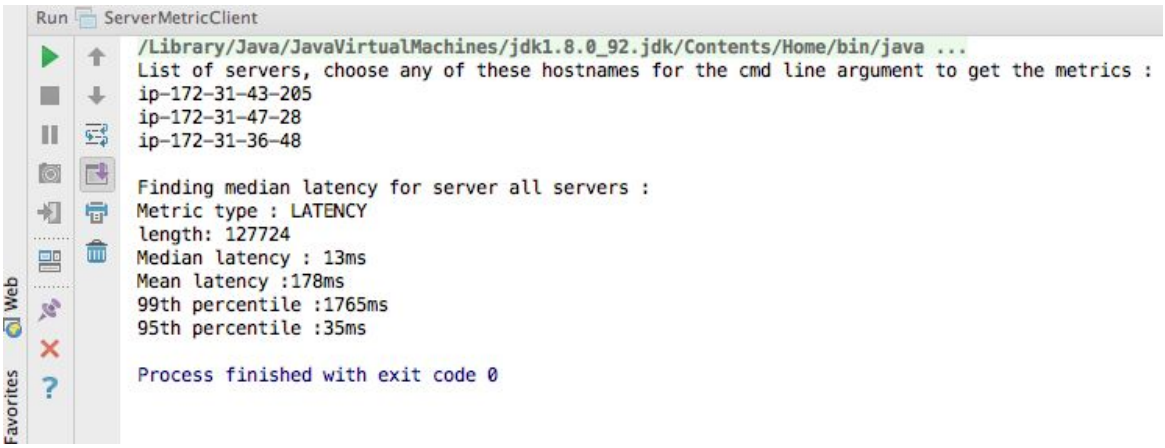


Step 3 : **Write(POST) Day 999**

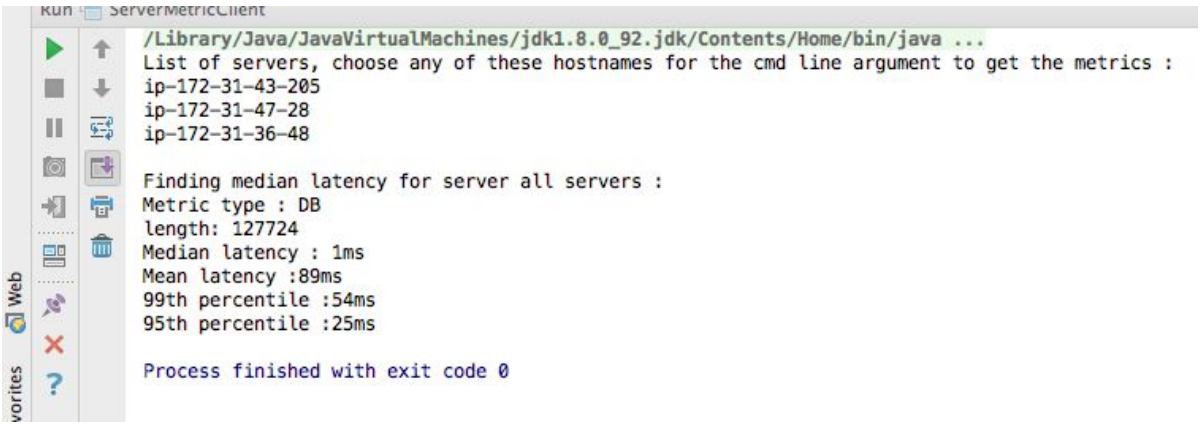
Client write metrics :



Server write metrics :

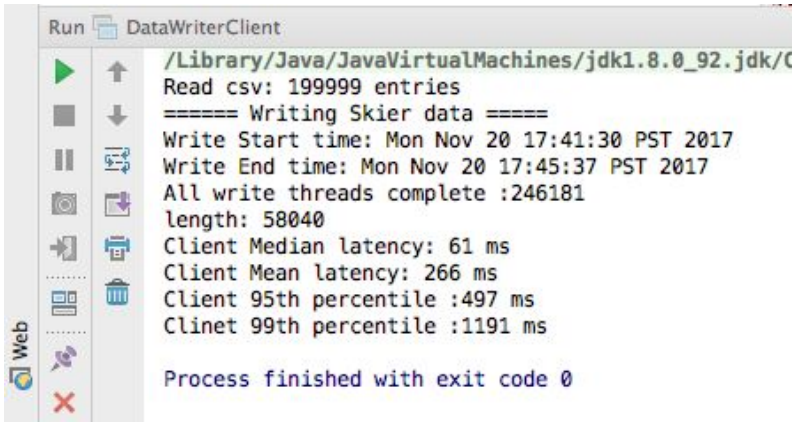


DB latency metrics :

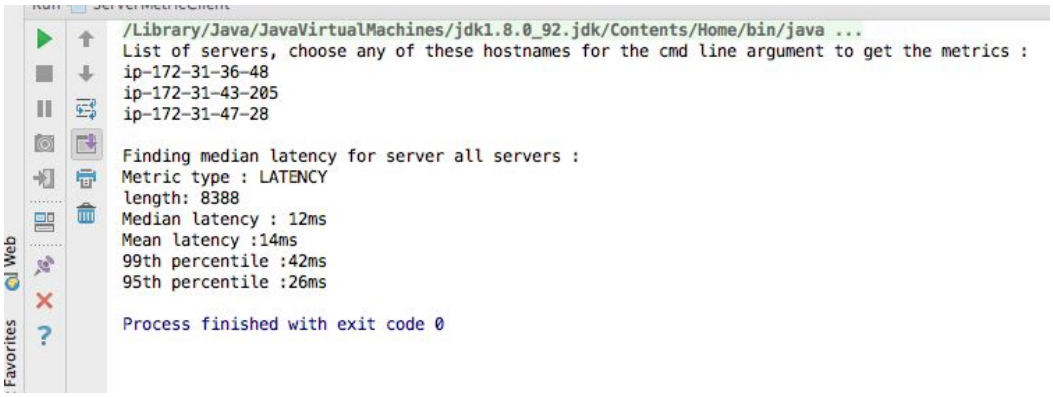


Data Write (POST) - Day 3

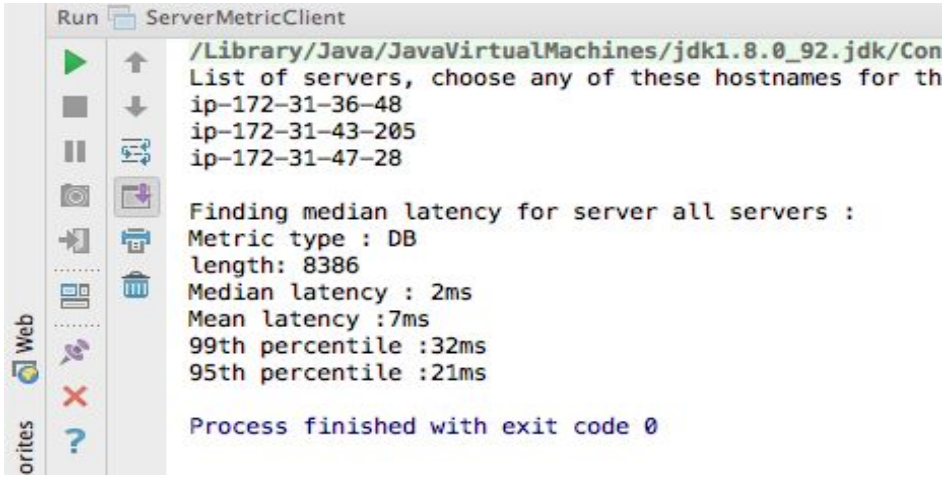
Client metrics:



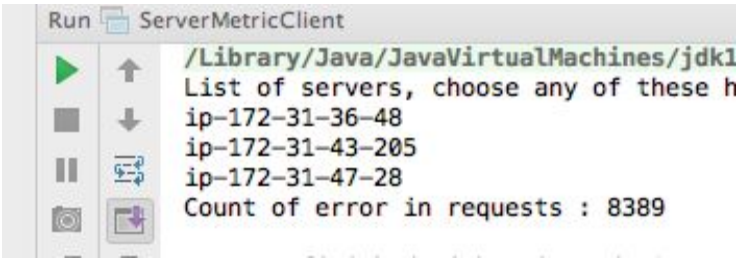
Server metrics :



DB Latency metrics:

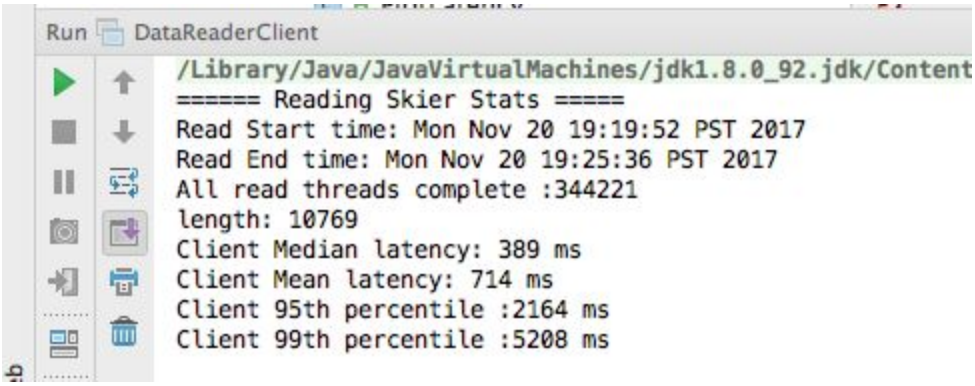


Error count :

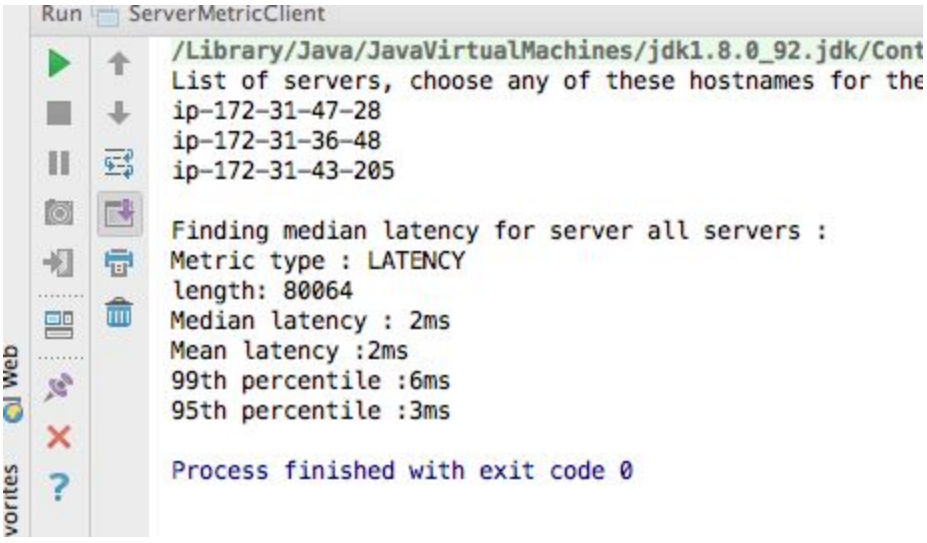


Read Day 3 Metrics :

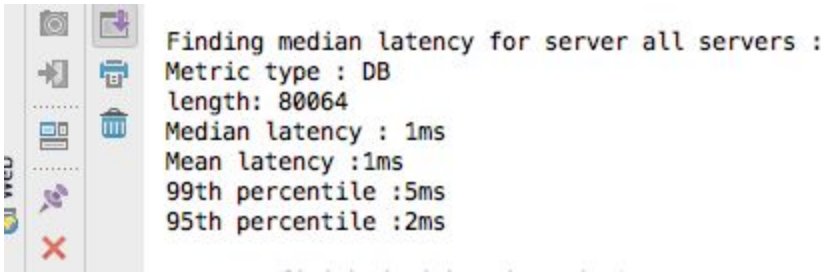
Client metrics :



Server metrics for latency :

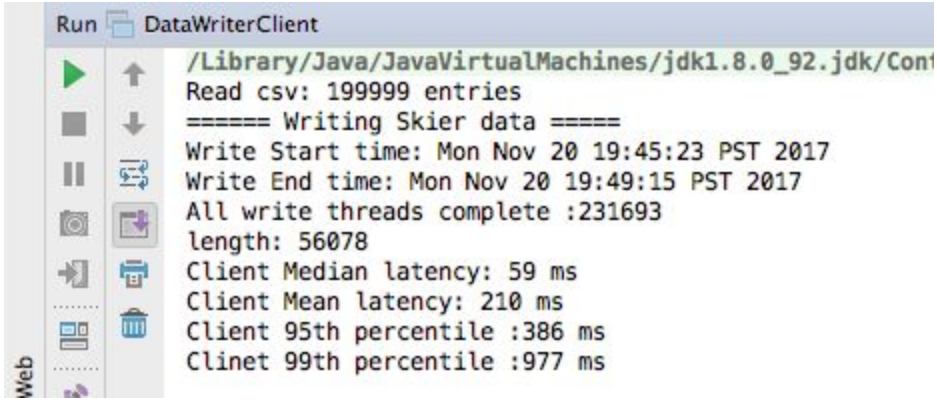


DB latency metrics :

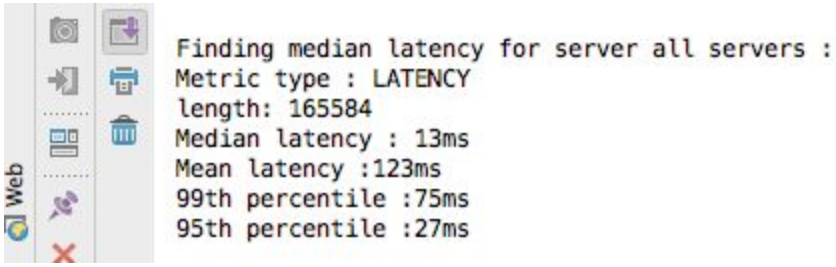


Day 4 - Write (POST) Metrics :

Client metrics :



Server metrics all hosts combined :

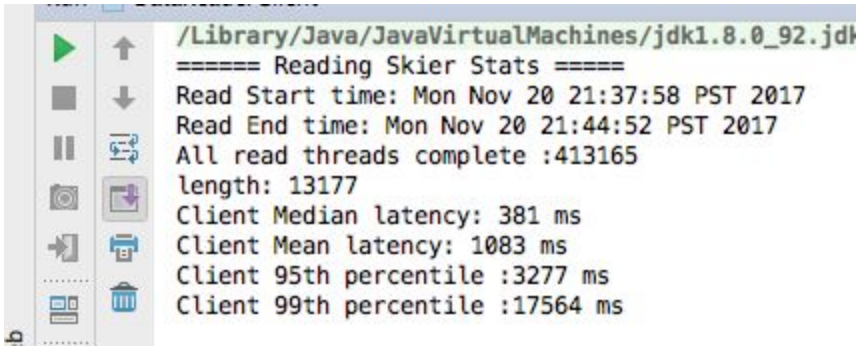


DB latency :

```
Finding median latency for server all servers :
Metric type : DB
length: 165584
Median latency : 2ms
Mean latency :62ms
99th percentile :36ms
95th percentile :22ms
```

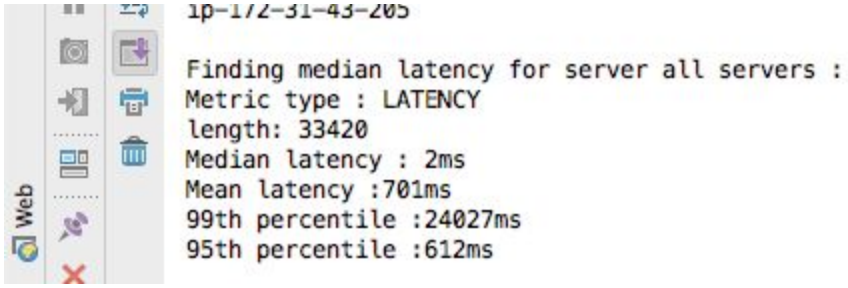
<u>Read Day 4 performance stats</u> :

Client metrics :



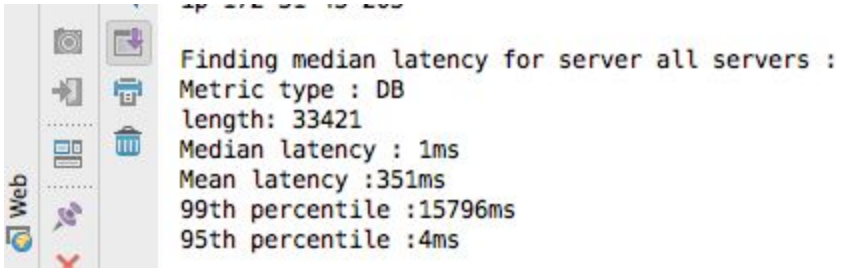
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_92.jdk
====== Reading Skier Stats =====
Read Start time: Mon Nov 20 21:37:58 PST 2017
Read End time: Mon Nov 20 21:44:52 PST 2017
All read threads complete :413165
length: 13177
Client Median latency: 381 ms
Client Mean latency: 1083 ms
Client 95th percentile :3277 ms
Client 99th percentile :17564 ms
```

Server metrics :



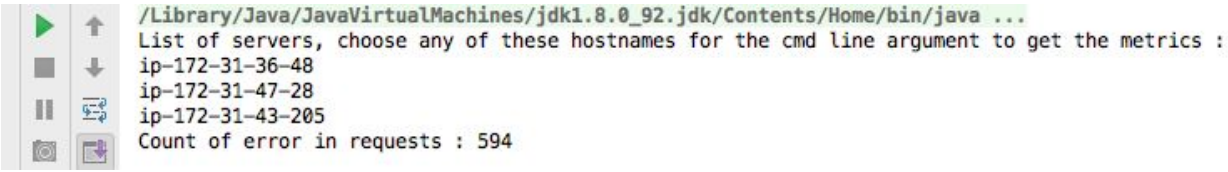
```
ip-172-31-43-205

Finding median latency for server all servers :
Metric type : LATENCY
length: 33420
Median latency : 2ms
Mean latency :701ms
99th percentile :24027ms
95th percentile :612ms
```

DB latency metrics:



```
Finding median latency for server all servers :
Metric type : DB
length: 33421
Median latency : 1ms
Mean latency :351ms
99th percentile :15796ms
95th percentile :4ms
```

Error count :



The reports suggests that the server metrics are clearly faster than the latencies reported by the client. With my design, the delay in writing the metrics in fixed intervals, this process would continue even after the read or write process is finished, could have used an external queue resource to avoid this slow reporting.