BSDS : cs6650 - HW 1

Github repo: https://github.com/skeerti2/cs6650

The homework 1 is inside this folder to accommodate future assignments in a single repo. I had requested Professor Ian for extension up to Sunday but was not able to complete by then due to a lot of unexpected errors and being stuck in Part 1 of Client for a lot of time. Hence, I would like to avail 2 late days (at 5% final grade) for Monday & Tuesday.

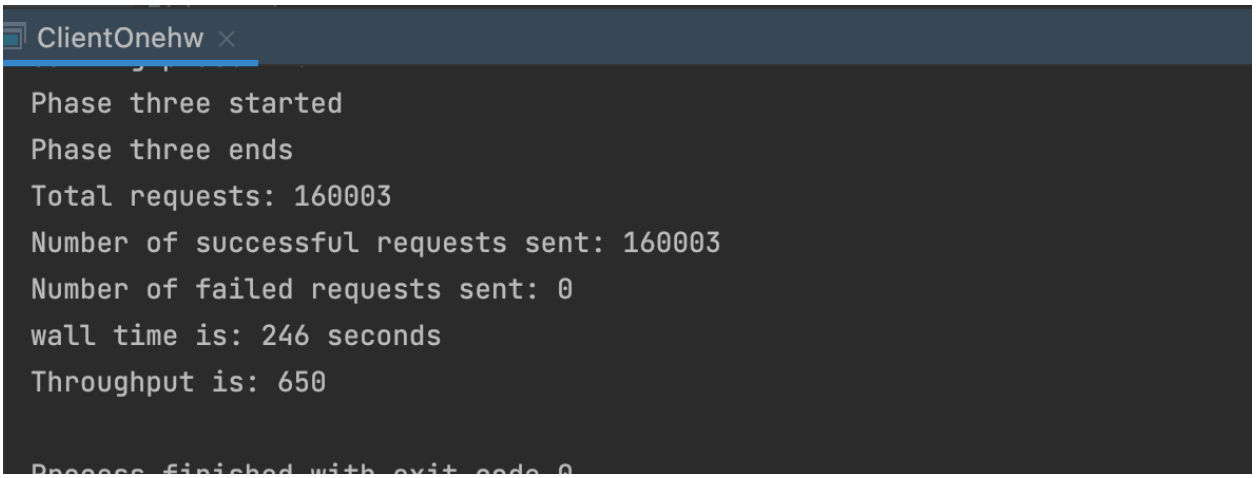
Client - 1: Design

For this Client, each of the methods doPhaseOne, doPhaseTwo and doPhaseThree are used to run the client in phases. Each phase has its own CountDownLatch in-order to avoid too many concurrent updates on a global countdown latch variable. Once the number of threads in phase one reaches 80% (since its counting down - meaning 20% threads have completed), doPhaseTwo would be called. For each phase the respective number of threads are created and within the anonymous function of the thread, a while loop is running to make required number of POST requests for each thread.

The response is inside a try-catch for phase 2 since I was getting too many connection time out errors that blocked the countdown latch for phase 2 (peak load). By using this, an exception is caught inside catch and while loop continues, ensuring that the required number of requests are made. There are 3 shared variable across all the three phases:
successfulRequests: To record successful requests across all phases. This is an Atomic integer to avoid race conditions.
unsuccessfulRequests: To record unsuccessful requests.
totalRequests: To record total requests, successful and unsuccessful.
On a bad response, the client sends request again 5 times until a successful request or gives up after that

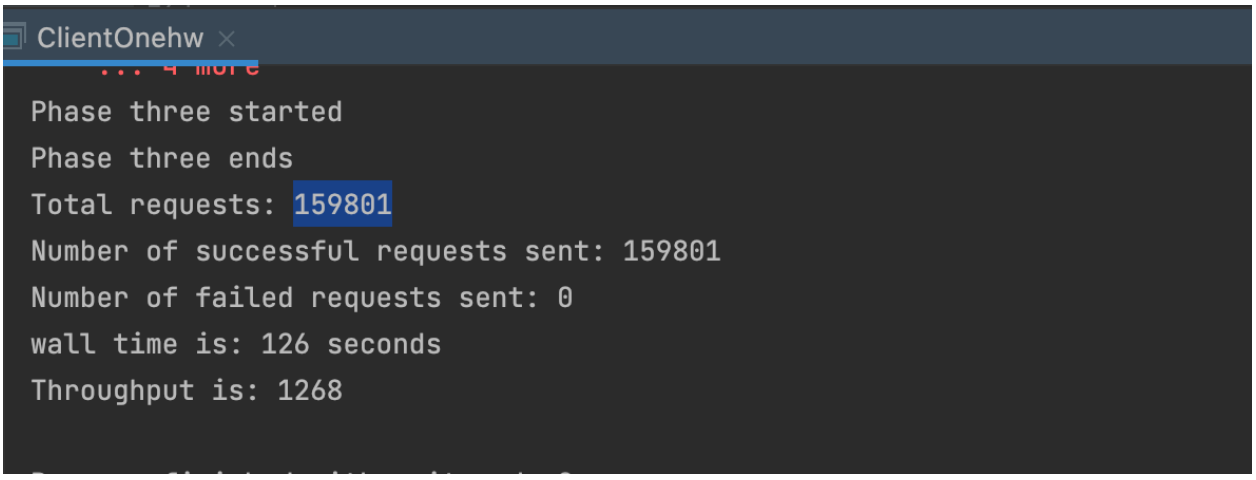
Client 2: Design

Client 2 is uses the same design, but records the time taken for each request to add to a synchronized arraylist - CopyOnWriteArrayList. This list is used to write data to csv.
Another separate list is used to store just the latency for each request so that median, mean etc can be easily computed. I figured that Apache commons Math library has methods to calculate median and percentile.



32 Threads: Client 1

64 Threads: Client 1



128 Threads: Client 1

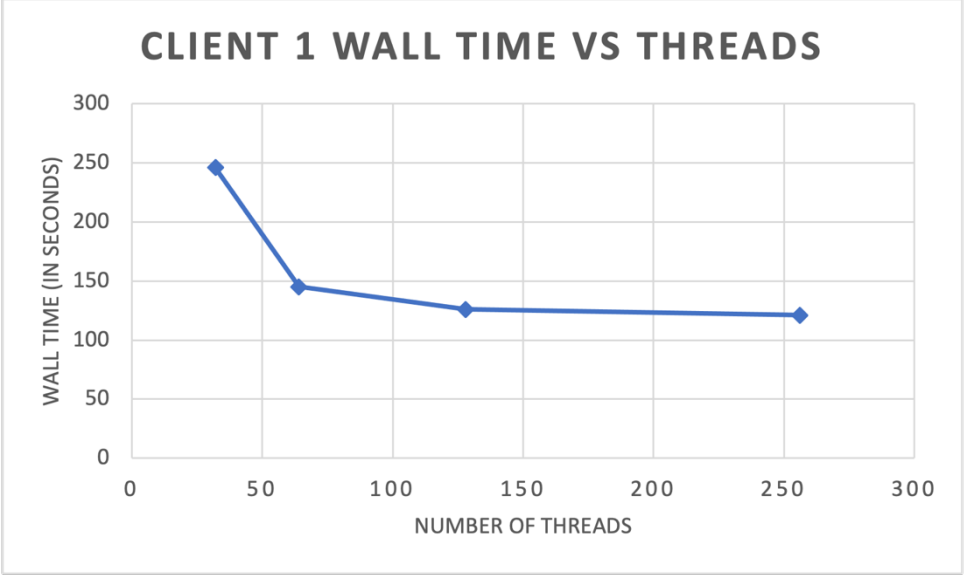
```
Total requests: 158615
Number of successful requests sent: 158615
Number of failed requests sent: 0
wall time is: 121 seconds
Throughput is: 1310

Process finished with exit code 0
```

256 Threads: Client 1

```
Number of successful requests sent: 159814
Number of failed requests sent: 0
wall time is: 20 seconds
Throughput is: 7990
Expected throughput for 10000 requests: 625

Process finished with exit code 0
```



```
Total requests: 160003
Number of successful requests sent: 160003
Number of failed requests sent: 0
wall time is: 357 seconds
Throughput is: 448
Mean response time is: 40 Milliseconds
Median is: 26.0
99th Percentile is: 320.0
Min latency is: 16
Max latency is: 8842
```

32 Threads: Client 2

```
Total requests: 159801
Number of successful requests sent: 159801
Number of failed requests sent: 0
wall time is: 343 seconds
Throughput is: 465
Mean response time is: 58
Median is: 27.0
99th Percentile is: 477.9800000000105
Min latency is: 17
Max latency is: 50956
```

64 Threads: Client 2

```
Total requests: 156313
Number of successful requests sent: 156313
Number of failed requests sent: 0
wall time is: 266 seconds
Throughput is: 587
Mean response time is: 53 Milliseconds
Median is: 26.0
99th Percentile is: 456.0
Min latency is: 18
Max latency is: 13055
```

128 Threads: Client 2

```
Total requests: 143295
Number of successful requests sent: 143295
Number of failed requests sent: 0
wall time is: 244 seconds
Throughput is: 587
Mean response time is: 66 Milliseconds
Median is: 26.0
99th Percentile is: 815.0400000000081
Min latency is: 18
Max latency is: 23893
```

256 Threads: Client 2