ASSIGNMENT I

Ravi Voleti

Link To assignment:

https://gortonator.github.io/bsds-6650/assignments-2021/Assignment-1

Code Repository:

https://github.com/xelese/BSDS/tree/master/Assignment1

Server Description*:*

The server has been configured on an AWS EC2 instance with the following configuration:

- Amazon Linux 2
- T2 Micro
- Tomcat 8.5
- US-EAST-1 (N. Virginia)

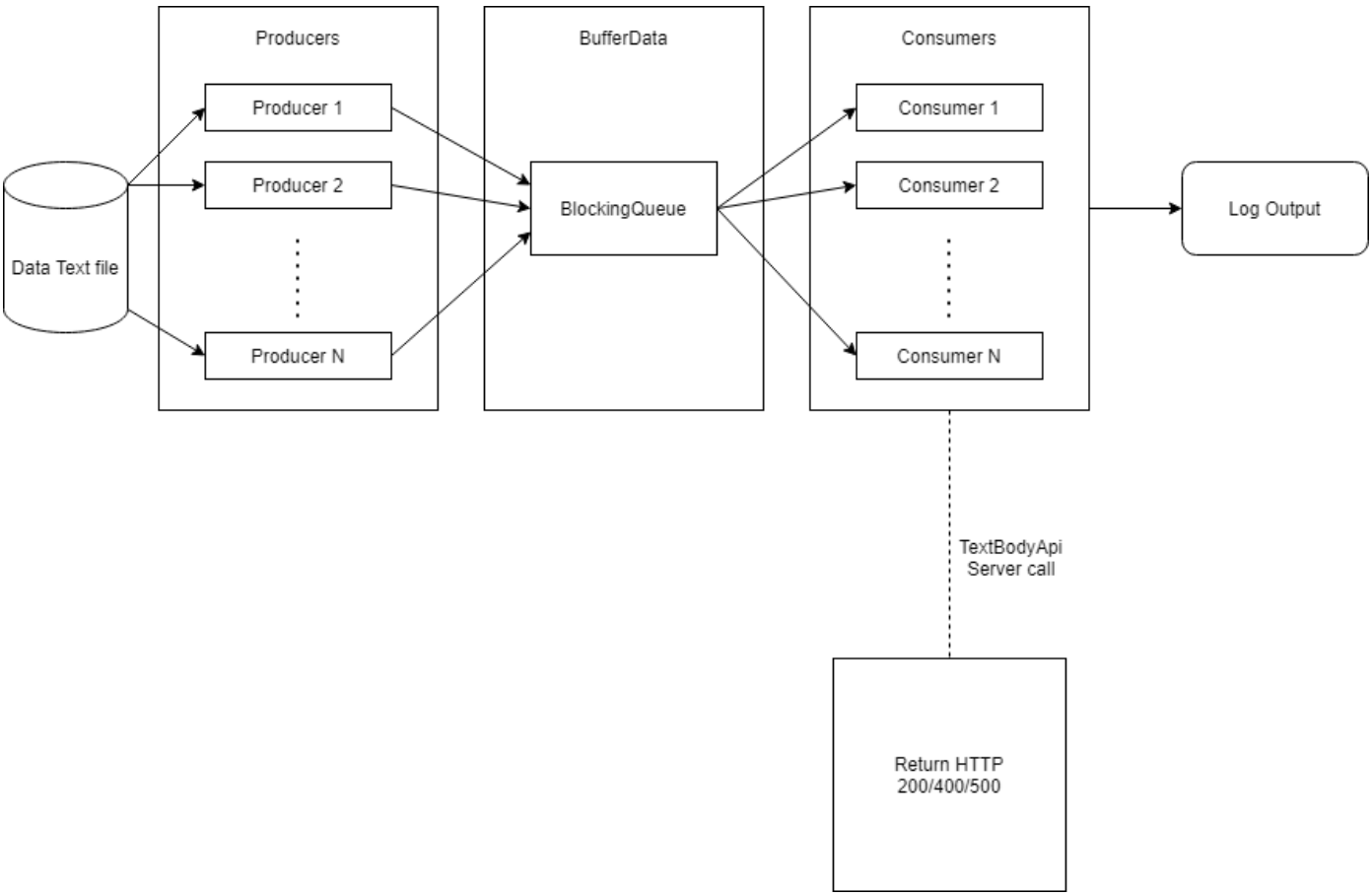
The server code was imported from Swagger running as a Spring boot java servlet deployed on the tomcat server.

Client Description*:*

- All Client code is written in Java utilizing the TextBodyApi from Swagger.
- Clients were run on a local machine in Boston with 500 Megabit connection.

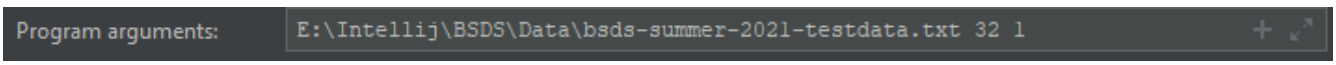
The execution flow of client is as follows:

I followed a simple Publisher Subscriber Design to solve the large data handling problem.



The code takes following inputs as shown below:

1. Data Text file
2. Max Consumer threads
3. Max Producer Threads



Overview of Execution:

Producer reads data line by line from the data fie if the line is not empty. To simplify data access, I have created a Data buffer class which holds all important data for execution. The data buffer class has a blocking queue <String> which is populated by the producer class. Each section of blocking queue is a single line of text.

As soon as data has been populated in the blocking queue, the consumer class pulls data from the queue and sends data to the server. All data received either error or a success message from the server is then logged in as output. For the logs, I have created an ArrayList<LoggingData> that stores the required results. At the end of the execution, the CustomLogger class calculates the results and publishes it on the terminal.

Threads and Management:

Both the Producer and the Consumer class implements the Runnable interface that enables us to submit the job to a thread pool and wait till all threads finish their execution. For managing the Thread pool itself, I took advantage of the Executor Service class and generated a fixed thread pool size based on user input. This way I can modify the total number of threads required by the program for each run.

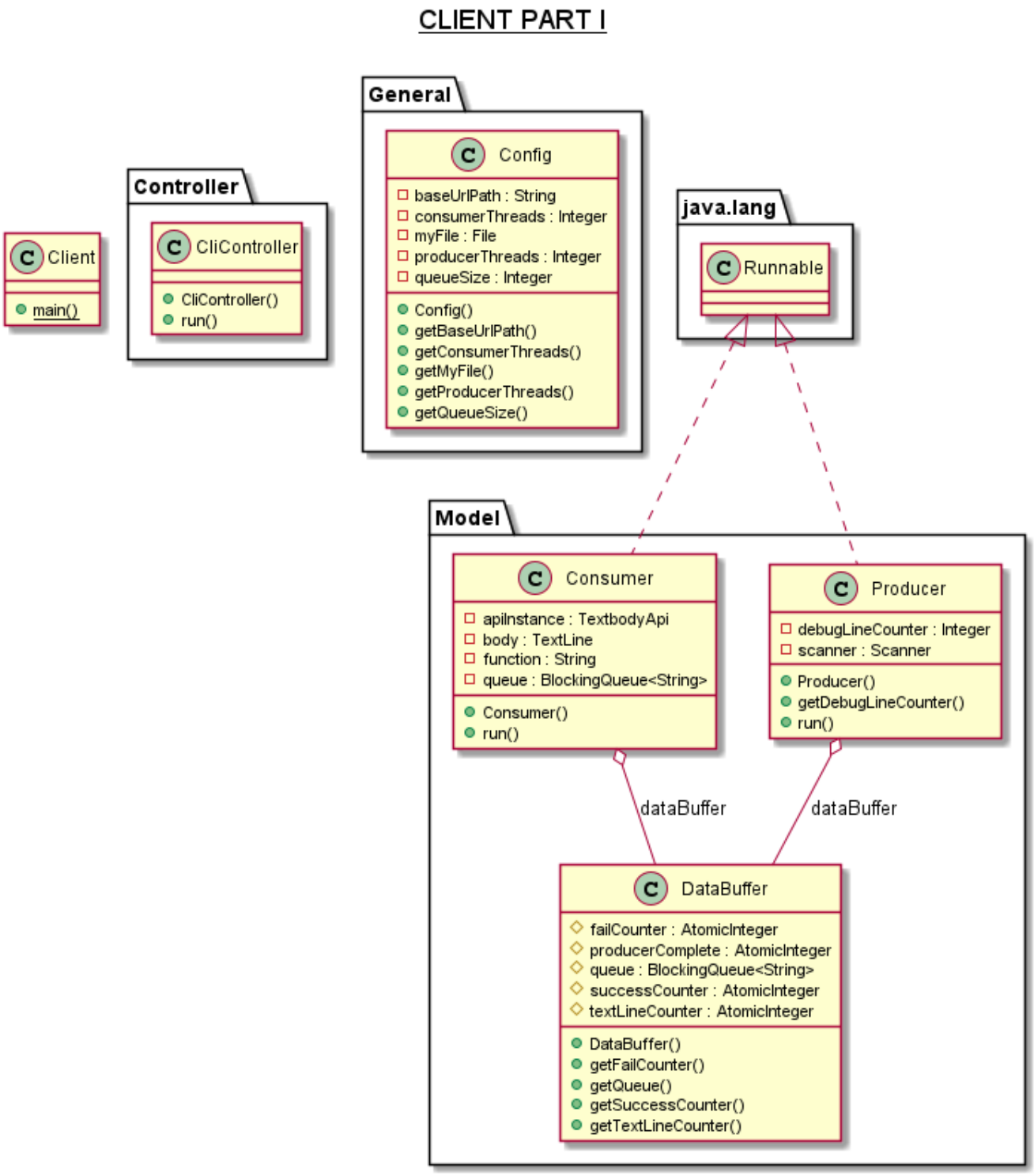
Safe termination:

I have added an atomic variable called *producerComplete* in the Data buffer class that keeps track of the total number of producers. At the end of producer's execution, it decrements this variable by 1. The consumer does not terminate its while loop execution until *producerComplete* <= 0 and blocking Queue is empty.

Another way to terminate safely was allowing the producer to send an end signal for the total number of consumers created and to stop their processing. Here is why I had not gone with that idea. Firstly, having multiple producers in this design does not allow us to write such code as this would force all the producers to send an end signal. This could be a problem if one of the producers sends the end signal and prematurely terminates all consumer threads. Additionally, only considering a single producer thread, in case of producer misbehavior or error we would not be able to terminate the queue safely as the end signal would never be populated in the BlockingQueue.

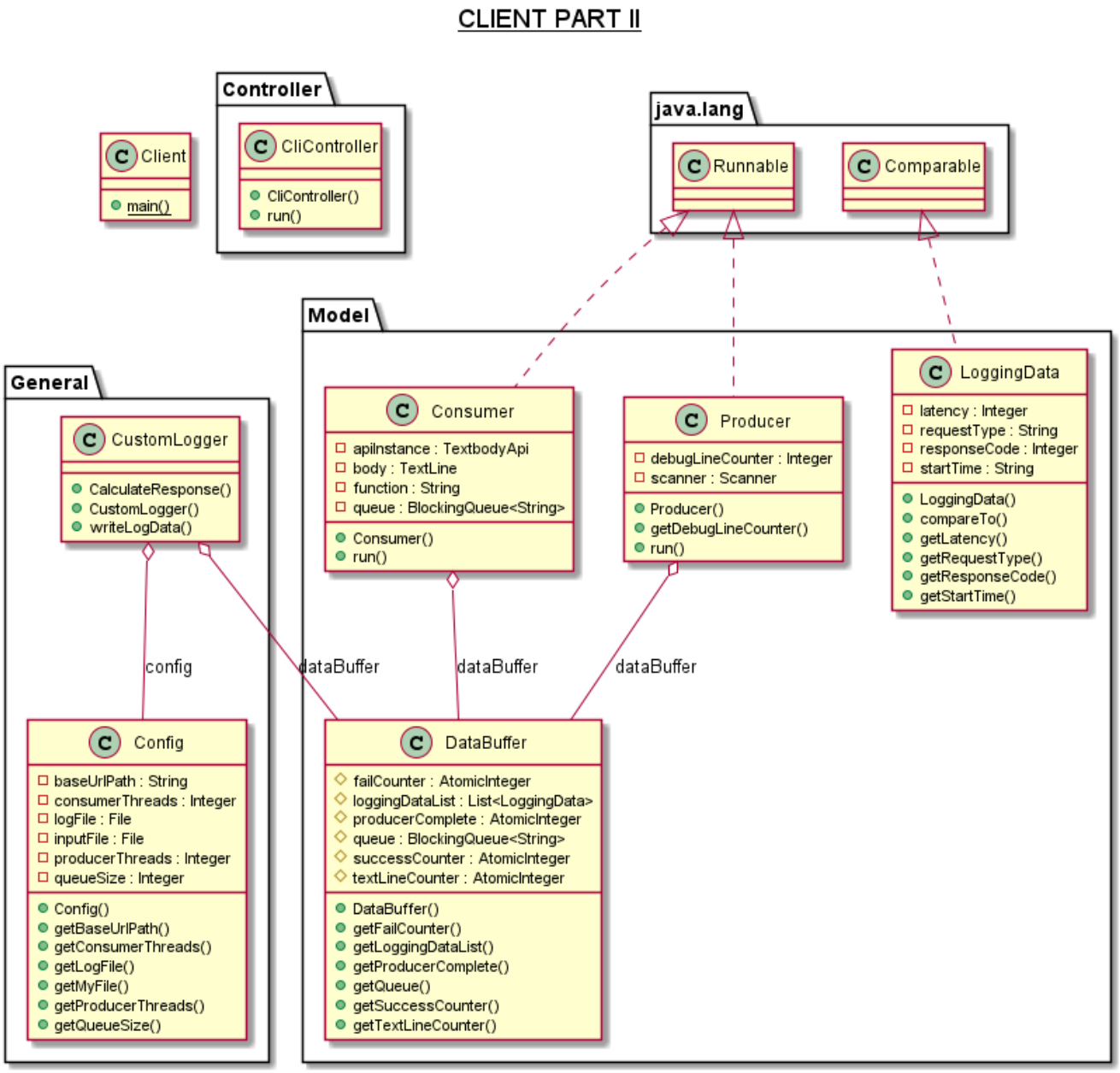
PART I:

Here is the class diagram for client part 1.



PART II:

Here is the class diagram for client part 2.



Here in extension of Part 1. I added a separate class called CustomLogger that takes data from DataBuffer class and outputs the log files within it. The CalculateResponse method within calculates the mean, median, maximum and the 90th Percentile for log data.

The LoggingData class contains the information (created in the Consumer) to be written into the log files. I had that class implement Comparable interface for additional sorting capabilities from the Collections class.

Client PART 1 RESULTS:

All configurations for Client part one are running with 1 producer thread, which reads contents form the text file 1 time. And N consumer threads. The client was run on the local machine while the server is hosted on an EC2 instance at us-east-1.

32 Threads:

```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 32
Total number of client requests consumed: 9930
------------------ OVERALL STATS -------------------
1. total number of successful requests sent: 9930
2. total number of unsuccessful requests sent: 0
3. wall time: 9.160842 s
4. Throughput: 1083.9615 req/s
----------------------------------------------------
```

64 Threads:

```
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 64
Total number of client requests consumed: 9930
------------------ OVERALL STATS -------------------
1. total number of successful requests sent: 9930
2. total number of unsuccessful requests sent: 0
3. wall time: 6.1110125 s
4. Throughput: 1624.9353 req/s
----------------------------------------------------
```

128 Threads:

```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 128
Total number of client requests consumed: 9930
------------------ OVERALL STATS -------------------
1. total number of successful requests sent: 9930
2. total number of unsuccessful requests sent: 0
3. wall time: 3.943144 s
4. Throughput: 2518.295 req/s
----------------------------------------------------
```

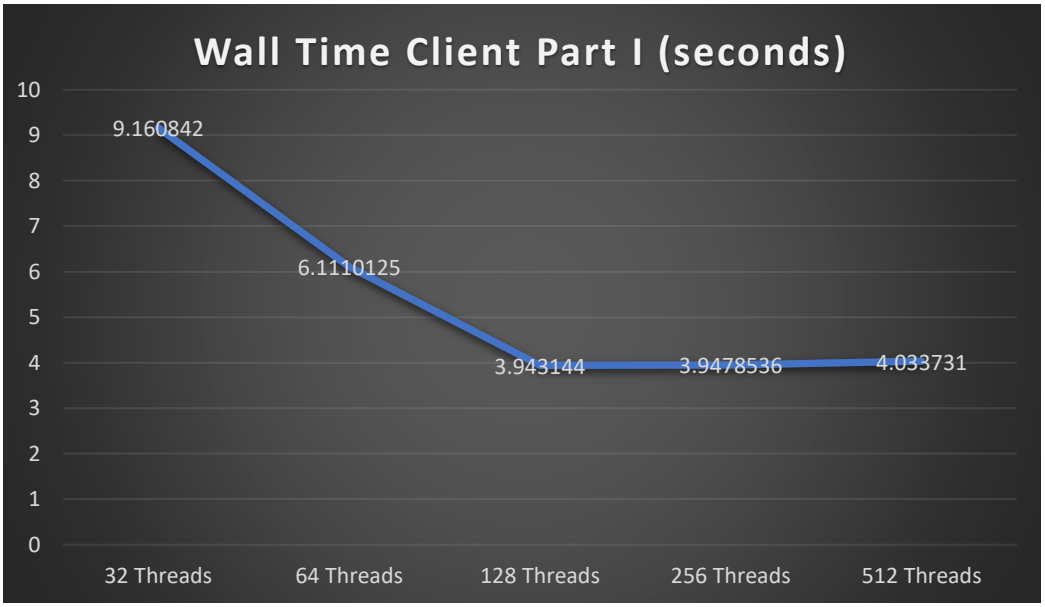
256 Threads:

```
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 256
Total number of client requests consumed: 9930
------------------ OVERALL STATS -------------------
1. total number of successful requests sent: 9930
2. total number of unsuccessful requests sent: 0
3. wall time: 3.9478536 s
4. Throughput: 2515.2908 req/s
----------------------------------------------------
```

512 Threads:

```
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 512
Total number of client requests consumed: 9930
------------------ OVERALL STATS -------------------
1. total number of successful requests sent: 9930
2. total number of unsuccessful requests sent: 0
3. wall time: 4.033731 s
4. Throughput: 2461.7407 req/s
----------------------------------------------------
```

Threads Vs Wall Time (Client Part I):



Client PART 2 RESULTS:

Configuration for client part 2 are the same as part 1.

32 Threads:

```
All threads terminated.
------------------ CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 32
Total number of client requests consumed: 9930
Log File location: .\log\2021-05-29-08-46-43-log.csv
------------------ OVERALL STATS ------------------
1. Total number of successful requests sent: 9930
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 9.254374 s
4. Throughput: 1073.0062 req/s
------------------ DETAILED STATS -----------------
1. Mean Response Time: 29.232729 ms
1. Median Response Time: 27.0 ms
1. Max Response Time: 354.0 ms
1. 90th Percentile: 33.0 ms
---------------------------------------------------
```

64 Threads:

```
All threads terminated.
------------------ CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 64
Total number of client requests consumed: 9930
Log File location: .\log\2021-05-29-08-55-41-log.csv
------------------ OVERALL STATS ------------------
1. Total number of successful requests sent: 9930
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 6.33639 s
4. Throughput: 1567.1384 req/s
------------------ DETAILED STATS -----------------
1. Mean Response Time: 40.006546 ms
1. Median Response Time: 34.0 ms
1. Max Response Time: 395.0 ms
1. 90th Percentile: 57.0 ms
---------------------------------------------------
```

128 Threads:

```
All threads terminated.
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 128
Total number of client requests consumed: 9930
Log File location: .\log\2021-05-29-09-12-10-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 9930
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 4.107257 s
4. Throughput: 2417.672 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 51.670494 ms
1. Median Response Time: 43.0 ms
1. Max Response Time: 434.0 ms
1. 90th Percentile: 77.0 ms
--------------------------------------------------
```

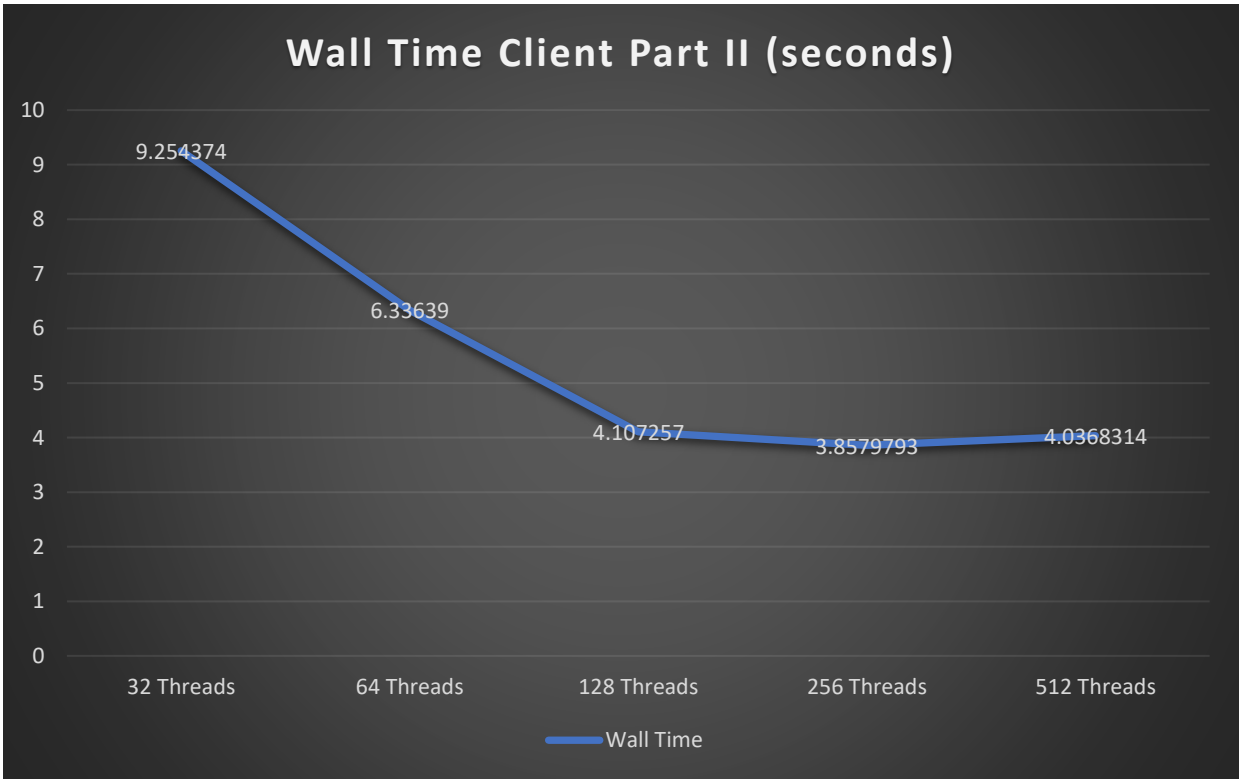
256 Threads:

```
All threads terminated.
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 256
Total number of client requests consumed: 9930
Log File location: .\log\2021-05-29-09-16-57-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 9930
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 3.8579793 s
4. Throughput: 2573.8862 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 96.26768 ms
1. Median Response Time: 80.0 ms
1. Max Response Time: 818.0 ms
1. 90th Percentile: 142.0 ms
--------------------------------------------------
```

512 Threads:

```
All threads terminated.
------------------ CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 512
Total number of client requests consumed: 9930
Log File location: .\log\2021-05-29-09-30-10-log.csv
------------------ OVERALL STATS ------------------
1. Total number of successful requests sent: 9930
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 4.0368314 s
4. Throughput: 2459.85 req/s
------------------ DETAILED STATS -----------------
1. Mean Response Time: 196.90312 ms
1. Median Response Time: 137.0 ms
1. Max Response Time: 1725.0 ms
1. 90th Percentile: 384.0 ms
---------------------------------------------------
```

Threads Vs Wall Time (Client Part II):



FUN TESTS (BONUS):

These tests were run to see how far the server can be pushed. In the following tests I increased the number of producers reading the file to gauge the maximum bandwidth capacity.

1024 Consumer threads, 10 Producer threads:

```
All threads terminated.
----------------- CONFIG --------------------------
Total number of producer Threads: 10
Total number of consumer Threads: 1024
Total number of client requests consumed: 99300
Log File location: .\log\2021-05-29-09-32-21-log.csv
----------------- OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 31.585941 s
4. Throughput: 3143.8037 req/s
----------------- DETAILED STATS ------------------
1. Mean Response Time: 318.23648 ms
1. Median Response Time: 287.0 ms
1. Max Response Time: 4344.0 ms
1. 90th Percentile: 437.0 ms
-------------------------------------------------
```

2048 Consumer threads, 10 Producer threads:

Results started throttling here, looks like I need to increase the number of producer threads to use all the 2048 threads efficiently.

```
All threads terminated.
----------------- CONFIG --------------------------
Total number of producer Threads: 10
Total number of consumer Threads: 2048
Total number of client requests consumed: 99300
Log File location: .\log\2021-05-29-09-50-04-log.csv
----------------- OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 35.816624 s
4. Throughput: 2772.4556 req/s
----------------- DETAILED STATS ------------------
1. Mean Response Time: 723.8354 ms
1. Median Response Time: 696.0 ms
1. Max Response Time: 6069.0 ms
1. 90th Percentile: 710.0 ms
-------------------------------------------------
```

3000 Consumer threads, 10 Producer threads:

Server errored out with connection timeout. Looks like this is the limit for a T2 micro EC2 Instance.