Assignment 1

<u>Client Design:</u>

1. Packages & Classes:

   A. part1

   - SkierClient.java:

   SkierClient is the major client class. The major things that this class do are:

   1. Check the validity of parameters.

   2. Generate the thread pools for all 3 phases. The time of starting different phases is controlled by using the countdown latches. All threads are added to a queue so that each thread can be retrieved later for the reports of part 1 and part 2.

   3. After all the threads finished their job, the report for part 1 will be print out to the console.

   4. The MetricsReporter will then be initialized for calculating data for part 2.

   - SkierThread.java:

   SkierThread is the class for each thread. The main jobs of this class are:
   1. Send designated number of POST request
   2. Send designated number of GET request
   3. Record the number of success and failure request
   4. The status of each request are bing kept in a list of ResponseStat. The status of request(ie. start time, end time, http status code(ie. 404), request method(ie.GET)) are recorded in class ResponseStat.
   5. Helper methods (ie. randomly generate lift id) are also defined in this class.

   B. part2

   - MetricsReporter.java:

   MetricsReporter is the class mainly for the calculation and report of part 2. The main functionalities of this class are:
   1. Read data from all the threads and requests by iterating the queue of threads from SkierClient.
   2. Calculate all the required metrics in part 2.
   3. Print put the report to the console and write out the record containing cvs file.

   - ResponseStat.java:

   ResponseStat is the class that record the status of each request (ie. start time, end time, http status code(ie. 404), request method(ie.GET)). Those information will be used by MetricsReporter.java for Part 2 report.

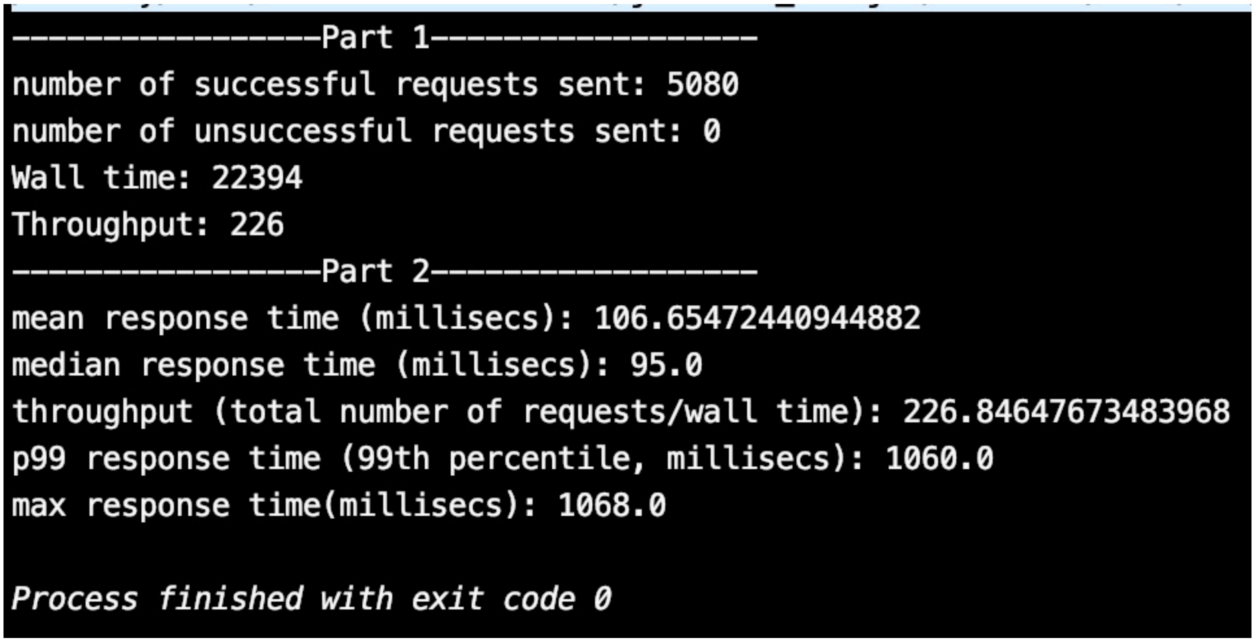
2. Class relationship

SkierClient Initialize SkierThread.
SkierThread send http requests and store request status in ResponseStat.
MetricsReporter read data from ResponseStat, and analysis those data.

Client Part 1 & Part 2:

1. 32 threads:

Result Screenshot:



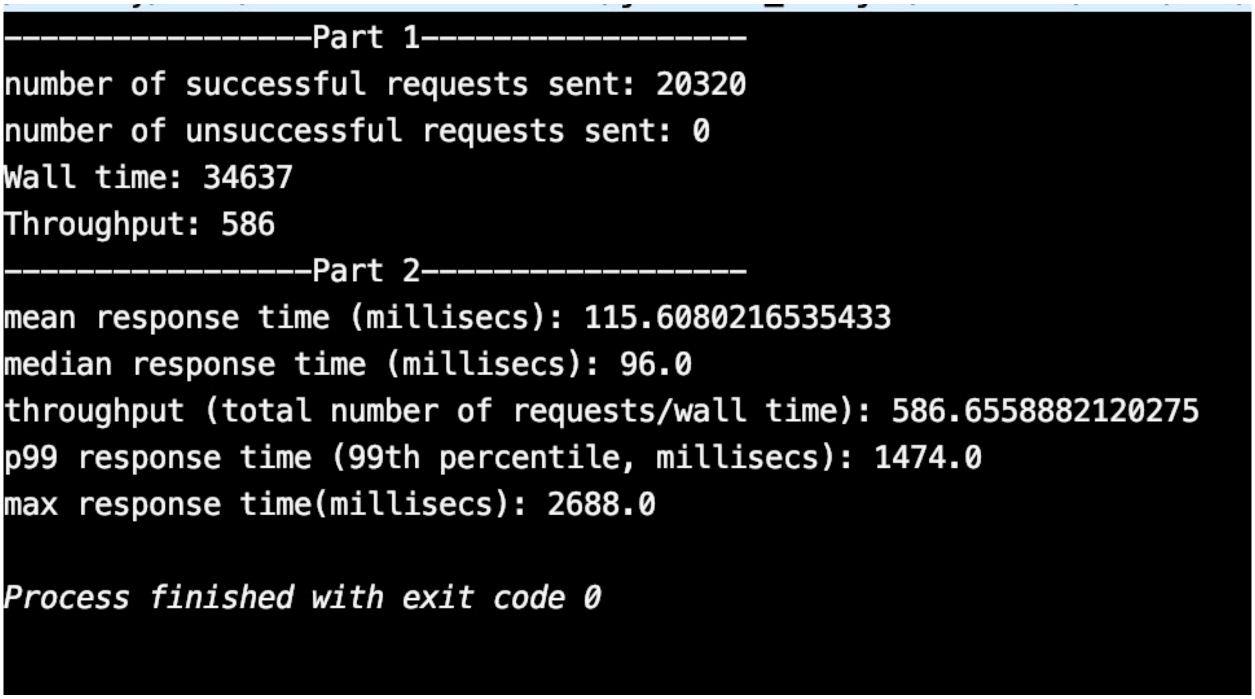
```
-----------------Part 1-----------------
number of successful requests sent: 5080
number of unsuccessful requests sent: 0
Wall time: 22394
Throughput: 226
-----------------Part 2-----------------
mean response time (millisecs): 106.65472440944882
median response time (millisecs): 95.0
throughput (total number of requests/wall time): 226.84647673483968
p99 response time (99th percentile, millisecs): 1060.0
max response time(millisecs): 1068.0

Process finished with exit code 0
```

2. 64 threads:

Result Screenshot:

3. 128 threads:

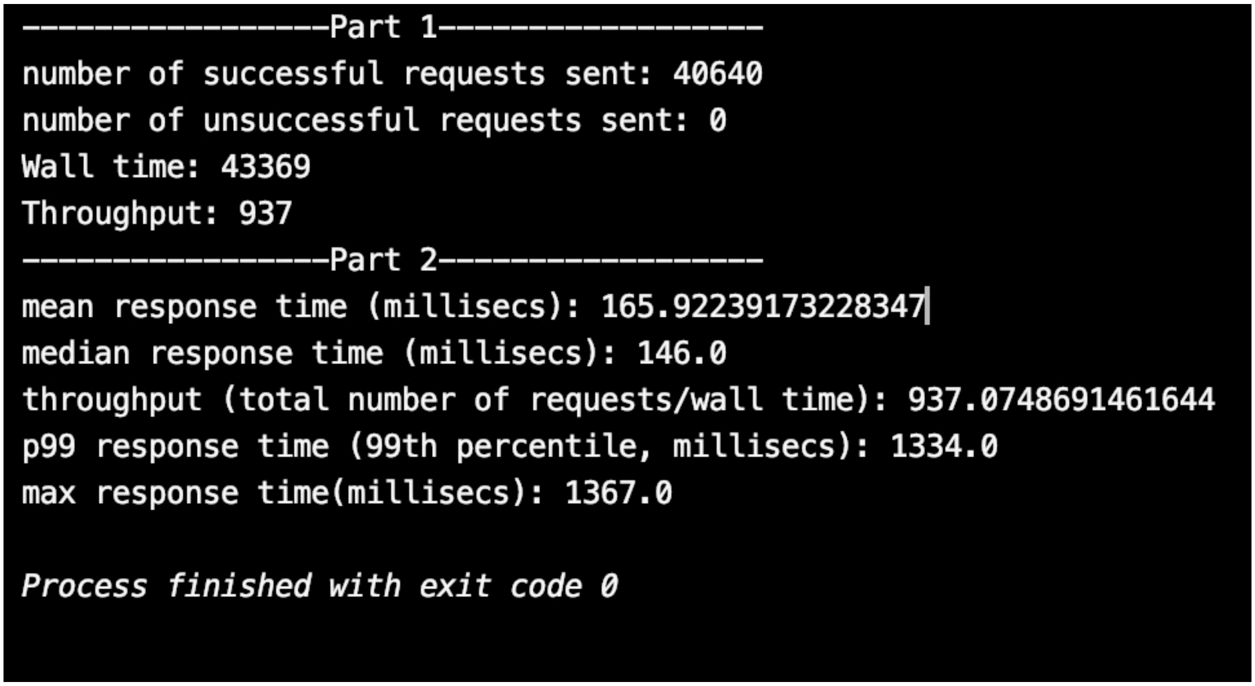
Result Screenshot:



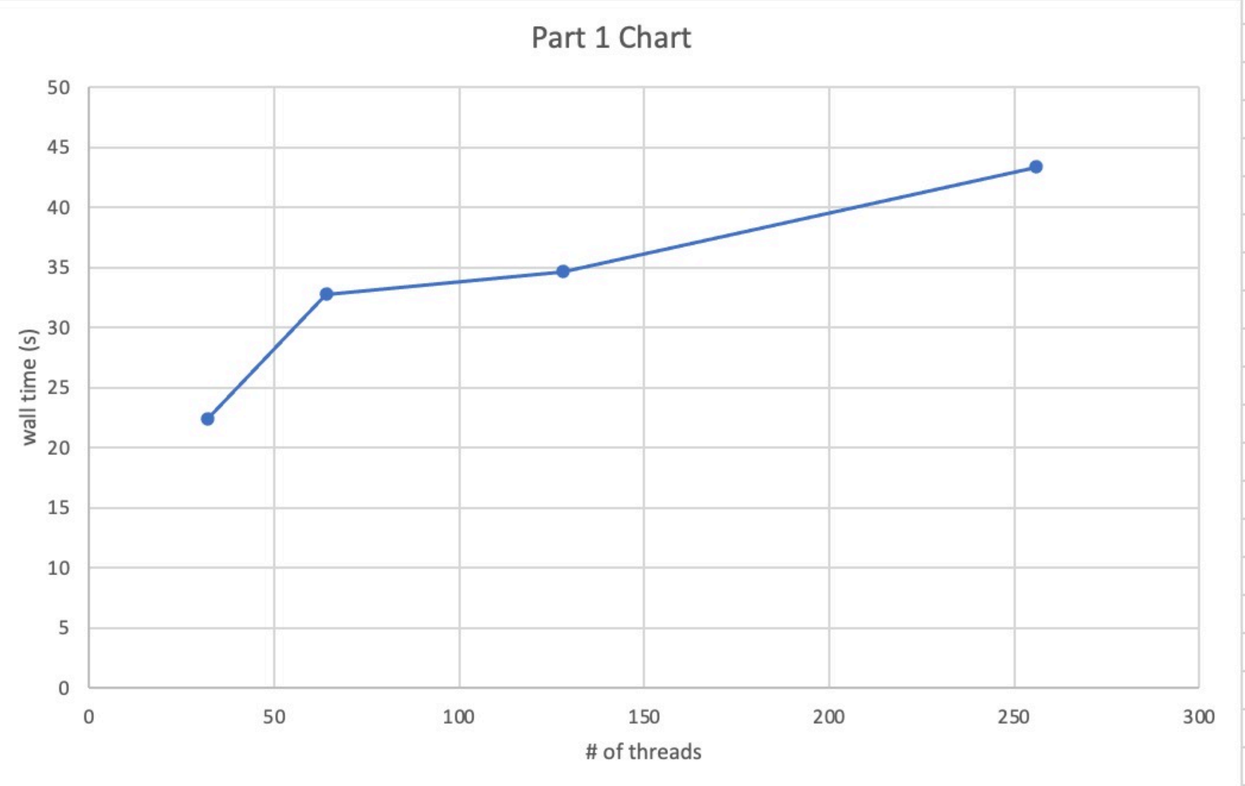
```
-----------------Part 1------------------
number of successful requests sent: 20320
number of unsuccessful requests sent: 0
Wall time: 34637
Throughput: 586
-----------------Part 2------------------
mean response time (millisecs): 115.6080216535433
median response time (millisecs): 96.0
throughput (total number of requests/wall time): 586.6558882120275
p99 response time (99th percentile, millisecs): 1474.0
max response time(millisecs): 2688.0

Process finished with exit code 0
```

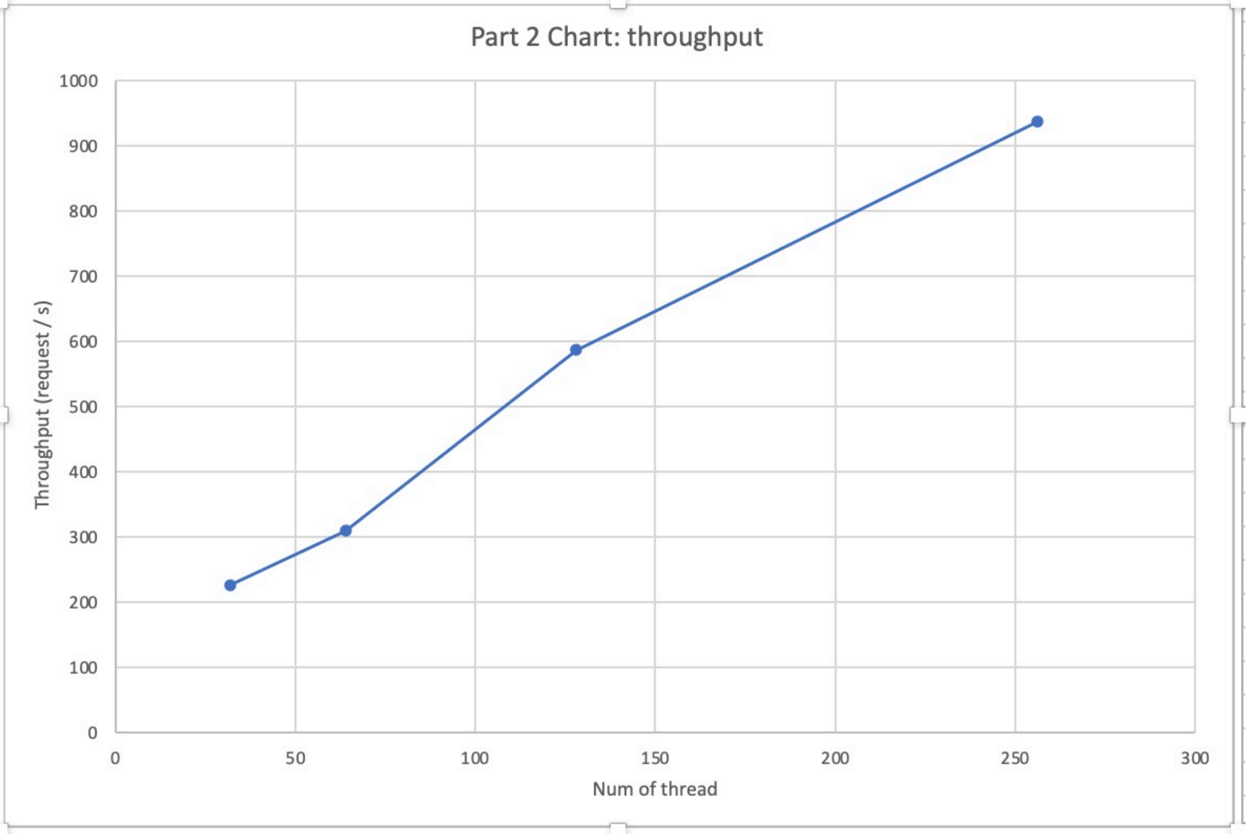
4. 256 threads:

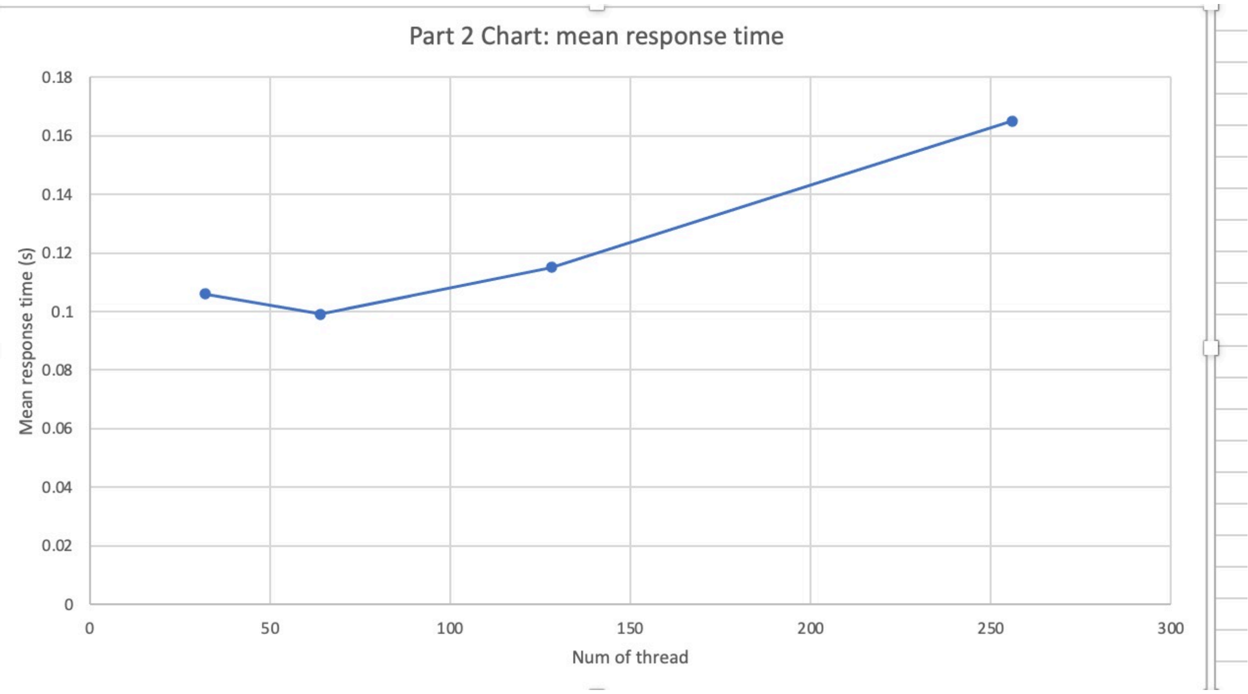
Result Screenshot:



```
-----------------Part 1------------------
number of successful requests sent: 40640
number of unsuccessful requests sent: 0
Wall time: 43369
Throughput: 937
-----------------Part 2------------------
mean response time (millisecs): 165.92239173228347
median response time (millisecs): 146.0
throughput (total number of requests/wall time): 937.0748691461644
p99 response time (99th percentile, millisecs): 1334.0
max response time(millisecs): 1367.0

Process finished with exit code 0
```


Part 1 Chart
wall time (s)
50
45
40
35
30
25
20
15
10
5
0
0
50
100
150
200
250
300
# of threads

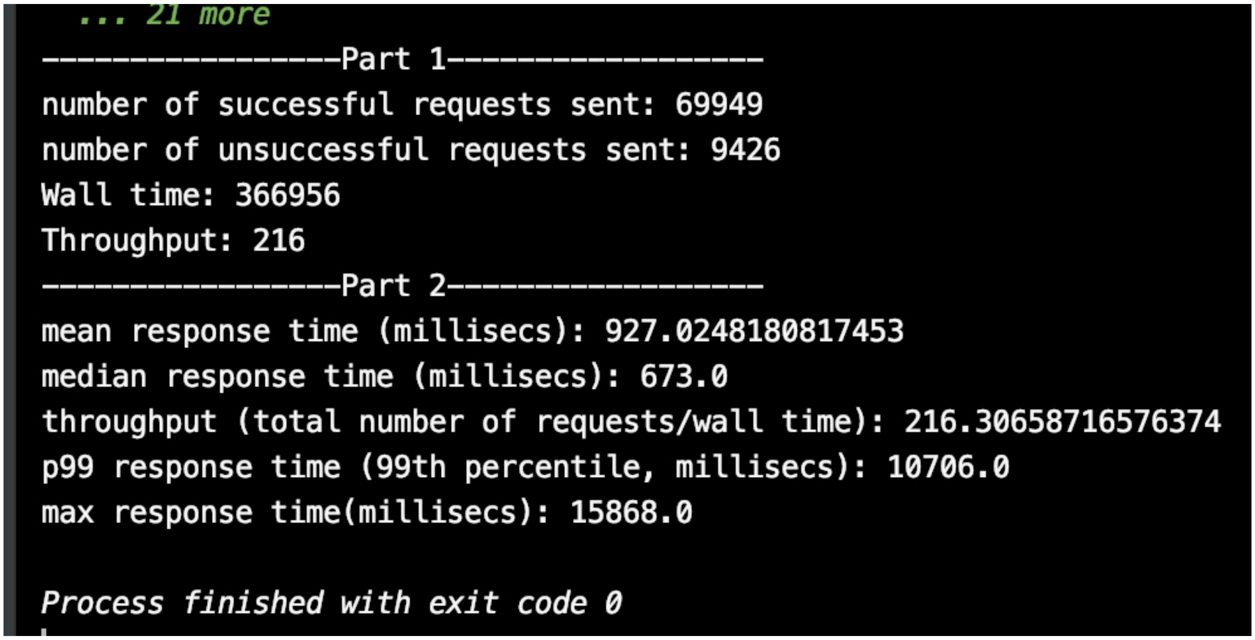

Part 2 Chart: throughput
Throughput (request / s)
1000
900
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
Num of thread



Bonus: Breaking things:

Observations:

In this test, I run 500 threads and the other numbers are kept same. In the screenshot we can see that there is a lot of failed request in this stress test. Most of those failed request are due to timeout.



```
  ... 21 more
-----------------Part 1-----------------
number of successful requests sent: 69949
number of unsuccessful requests sent: 9426
Wall time: 366956
Throughput: 216
-----------------Part 2-----------------
mean response time (millisecs): 927.0248180817453
median response time (millisecs): 673.0
throughput (total number of requests/wall time): 216.30658716576374
p99 response time (99th percentile, millisecs): 10706.0
max response time(millisecs): 15868.0

Process finished with exit code 0
```