cs6650_assignment1_QingChen

email: chen.qing1@northeastern.edu

- client_link: folders client1 and client2 included
  - https://github.khoury.northeastern.edu/chenqing/cs6650_assignment1_client
- server_link: src/main/java/SkierServlet.java
  - https://github.khoury.northeastern.edu/chenqing/cs6650_assignment1_server
- App name: `cs6650_lab2_war_exploded`
- port: `8080`
- ec2链接 http://35.89.205.211:8080/cs6650-lab2_war exploded/skiers/12/seasons/2019/day/1/skier/123
- local链接： http://localhost:8080/cs6650_lab2_war_exploded/skiers/12/seasons/2019/day/1/skier/123

before running client, make sure maven dependencies loaded and Tomcat services started in server folder

1. Build the Client (Part 1)

Client 1 could be found by the path below:

```
src/main/java/client1/SkierLiftRideClient.java
```

at line 18, you can choose local url or remote url for EC2

```
private static final String BASE_PATH =
  "http://localhost:8080/cs6650_lab2_war_exploded";//local path
private static final String BASE_PATH =
  "http://35.89.205.211:8080/cs6650-lab2_war%20exploded/"; replace ec2 ip address
```

- run main method to send requests to the server.

For the number of threads used after the first 32 threads, could try different solution by choosing from line 67.

```
int remainingEvents = NUM_EVENTS - (NUM_THREAD * NUM_POSTS_THREAD);
// int newNumOfRequest = 1680; // 100 threads
// int newNumOfRequest = 3360; // 50 threads
int newNumOfRequest = 1000; // 168 threads
```

2. Build the Client (Part 2)

Client 2 could be found by the path below:

```
src/main/java/client2/SkierLiftRideClient2.java
```

at line 24, you can choose local url or remote url for EC2

```
private static final String BASE_PATH =
  "http://localhost:8080/cs6650_lab2_war_exploded";//local path
private static final String BASE_PATH =
  "http://35.89.205.211:8080/cs6650-lab2_war%20exploded/"; replace ec2 ip address
```

run main method to send requests to the server.

For the number of threads used after the first 32 threads, could try different solution by choosing from line 74.

```
int remainingEvents = NUM_EVENTS - (NUM_THREAD * NUM_POSTS_THREAD);
int newNumOfRequest = 1680; // 100 threads
// int newNumOfRequest = 3360; // 50 threads
// int newNumOfRequest = 1000; // 168 threads
```

3. Structure of Clients

- `model / SkierLiftRideEvent.java` --> an object that generates a random skier lift ride event
  - getters / setters / constructors / toString()
- `model / SkierLiftRideEventList.java` --> store all SkierLiftRideEvent, includes **generator()** to generate random event
  - generator() / getters / setters / constructors / toString()
- `client1 / SkierLiftRideClient.java `--> generates threads and execute post requests

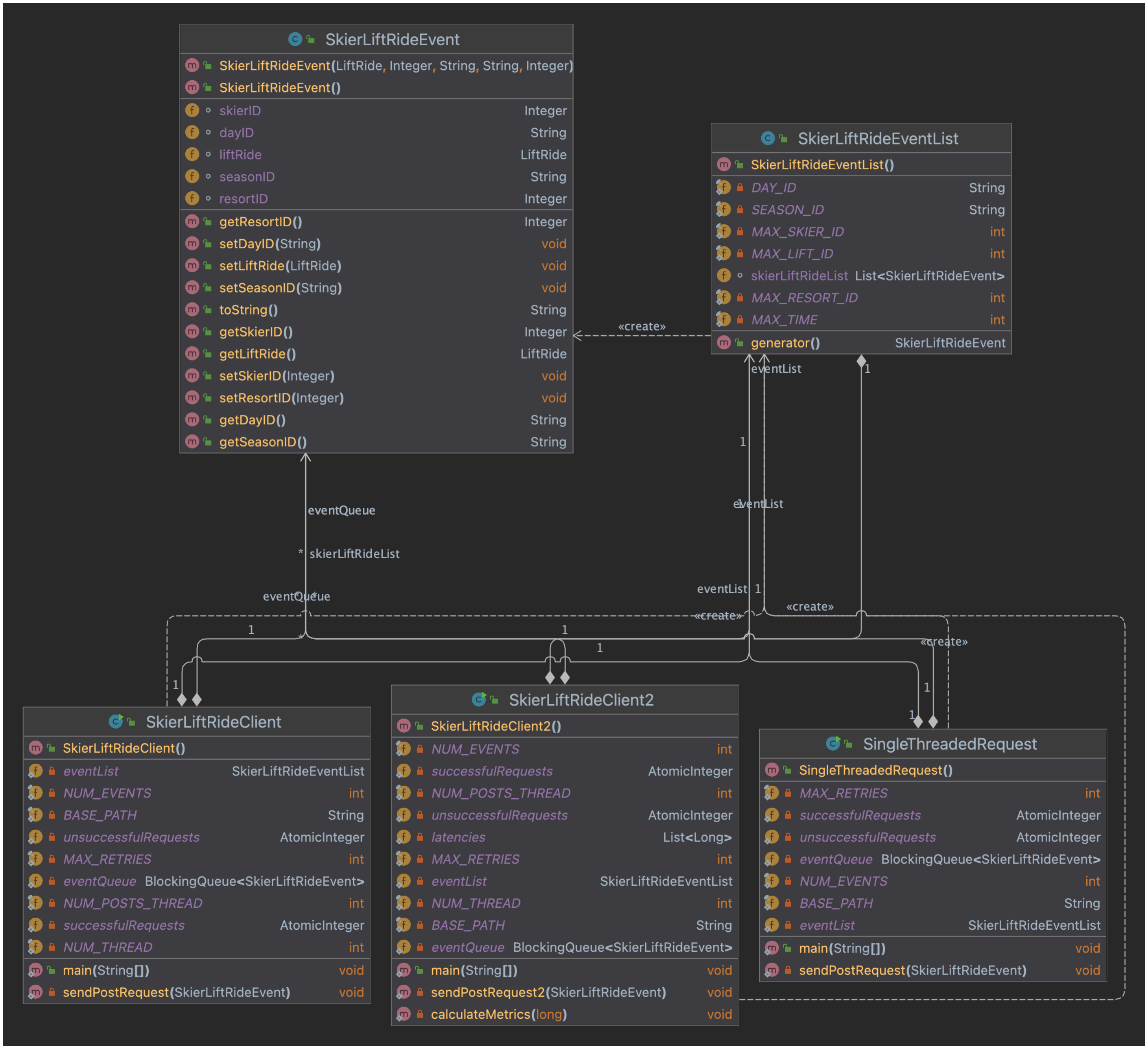
  - main() / sendPostRequest()

- `client2 / SkierLiftRideClient2.java` --> generates threads and execute post requests, calculate latencies.
  - main() / sendPostRequest() / calculateMetrics()
  - Wrote out a record: `client2responses.csv` under root directory

```
try (FileWriter writer = new FileWriter("client2responses.csv", true)) {
  writer.append(
    String.format("%d,%s,%d,%d\n", postStart, "POST", latency, responseCode)
  );
} catch (IOException e) {
  System.out.println("Failed to write to CSV file: " + e.getMessage());
}
```

clients will define the number of threads and SkierLiftRide events. In main(), will generate the specific amount of SkierLiftRide Objects, then send post requests by calling `writeNewLiftRide()`.

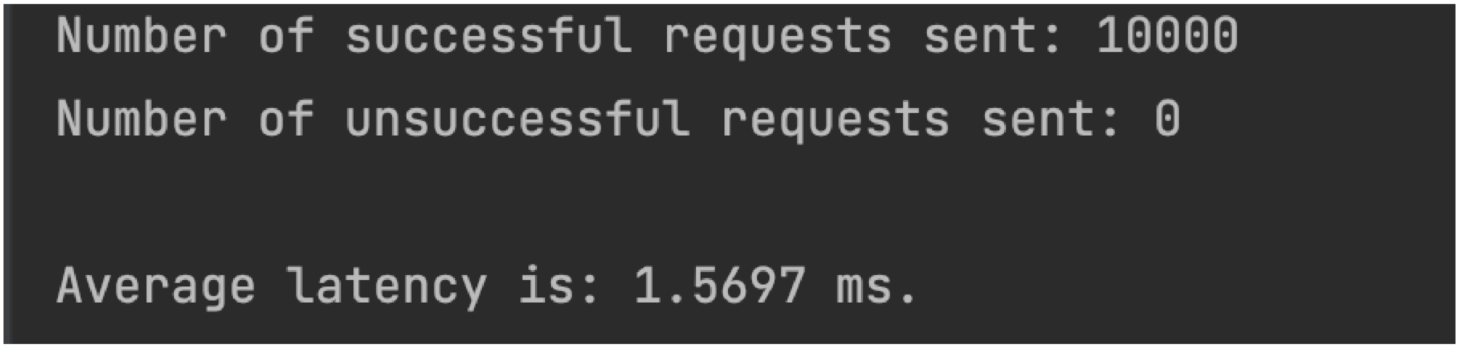
--- UML for your reference ---



4. Little's Law throughput predictions.

According to SingleThreadedRequest.java, we tested 10000 requests with only one thread, the average local request latency is 1.6ms.

local

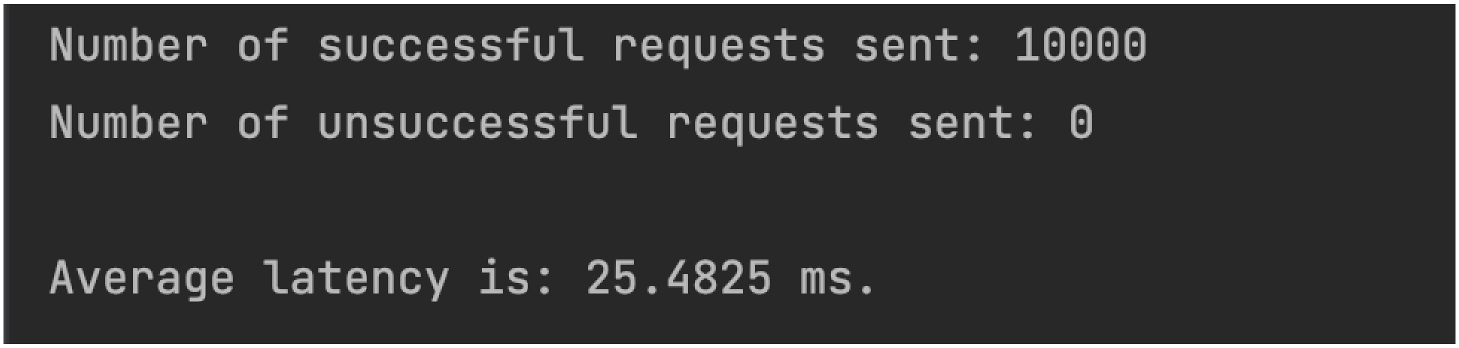


```
Number of successful requests sent: 10000
Number of unsuccessful requests sent: 0


Average latency is: 1.5697 ms.
```

now we use this data to predict throughput.

```
N=λW, numOfEvents = 200000

Startup: N = 32 threads W = 0.0016s --> λ = 32/0.0016 = 20000 requests/second

Remaining requests:

    * assumption1 N = 100 threads W = 0.0016s
    λ = 100 /0.0016 = 62500 requests/second
- total throughtput :  41250 requests/second

    * assumption2 N = 50 threads W = 0.0016s
    λ = 50 / 0.0016 = 31250 requests/second
- total throughtput : 25625 requests/second
```

ec2



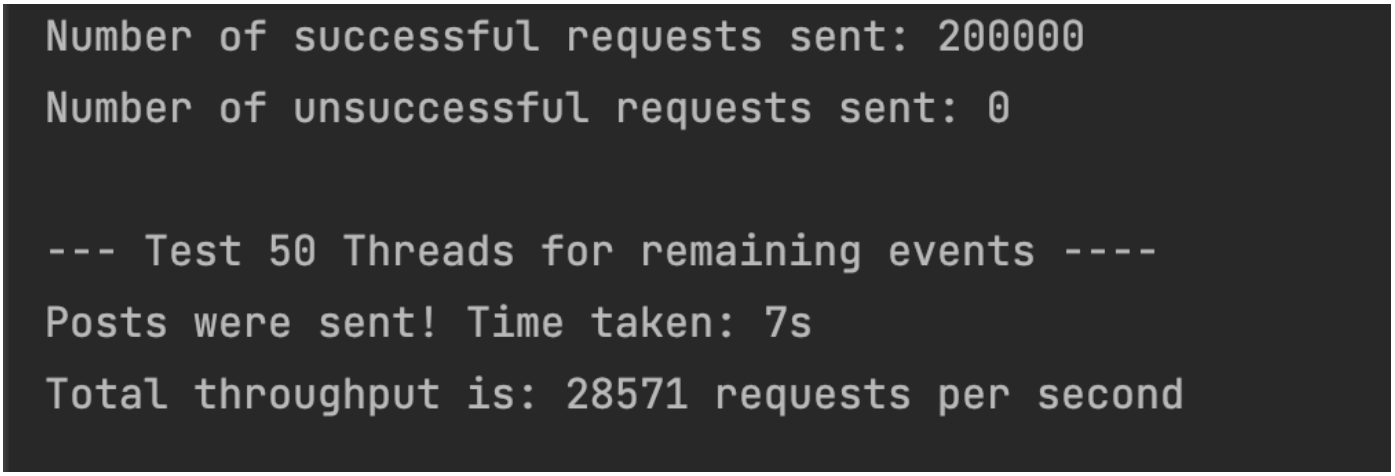
```
Number of successful requests sent: 10000
Number of unsuccessful requests sent: 0


Average latency is: 25.4825 ms.
```

now we use this data to predict throughput.

```
N=λW, numOfEvents = 200000

Startup: N = 32 threads W = 30ms --> λ = 32/3ms = 1000 requests/second

Remaining requests:
    * assumption2 N = 100 threads W = 0.0016s
    λ = 100 / 30ms = 3333 requests/second
- total throughtput : 2000 requests/second

    * assumption3 N = 50 threads W = 0.0016s
    λ = 50 / 30ms = 1667 requests/second
- total throughtput : 1300 requests/second
```

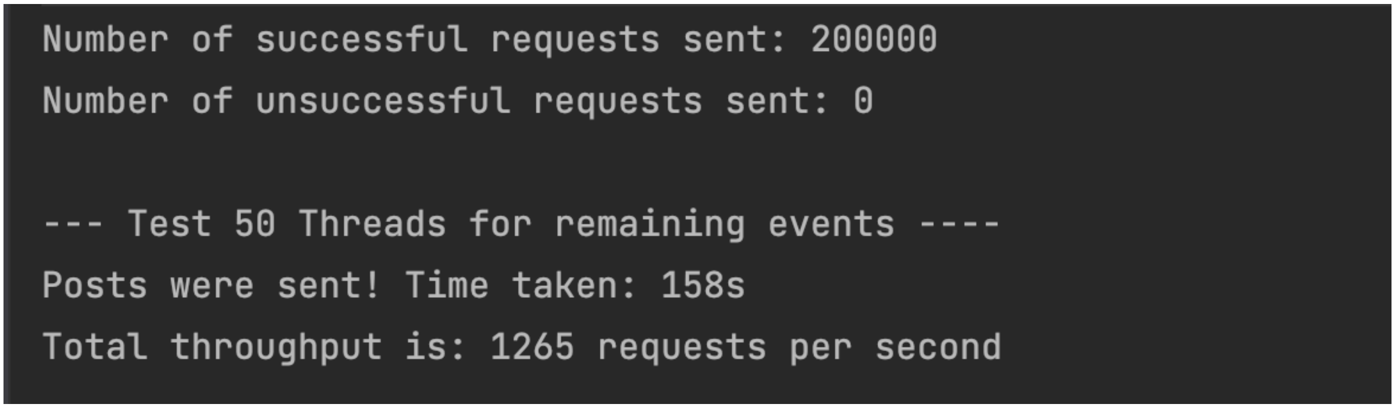
5. Actual throughput -- Client Part 1

- **local: created 50 threads for remaining requests:**



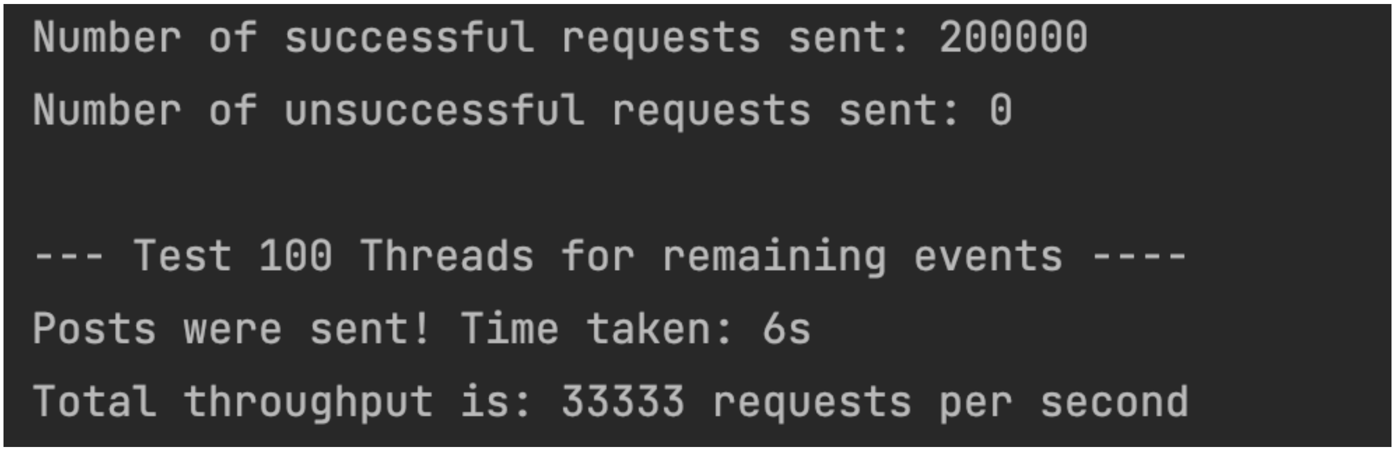
```
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 50 Threads for remaining events ----
Posts were sent! Time taken: 7s
Total throughput is: 28571 requests per second
```

- **ec2: created 50 threads for remaining requests:**



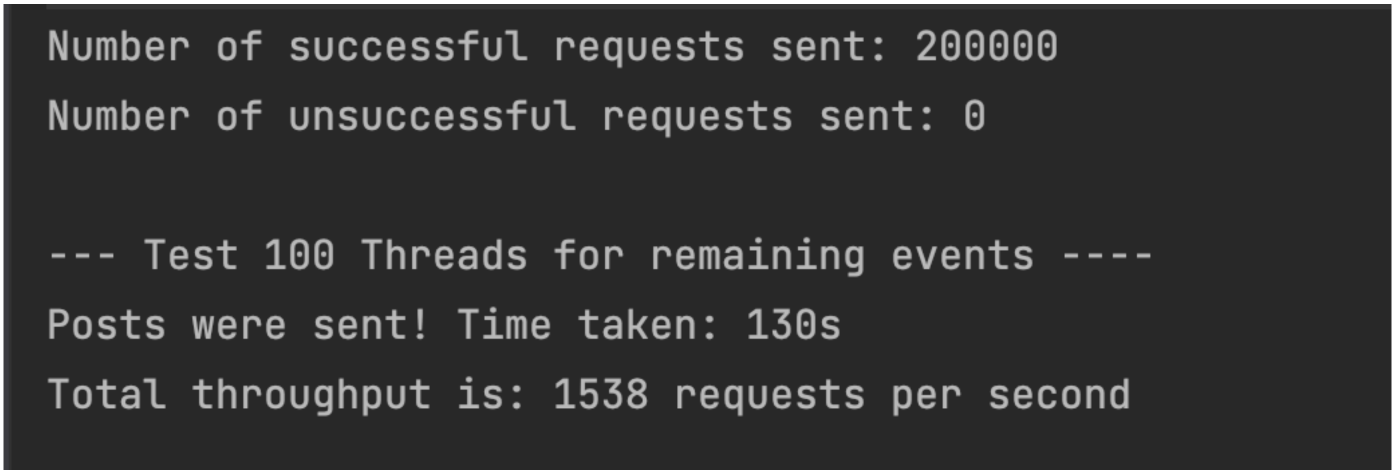
```
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 50 Threads for remaining events ----
Posts were sent! Time taken: 158s
Total throughput is: 1265 requests per second
```

- **local: created 100 threads for remaining requests:**



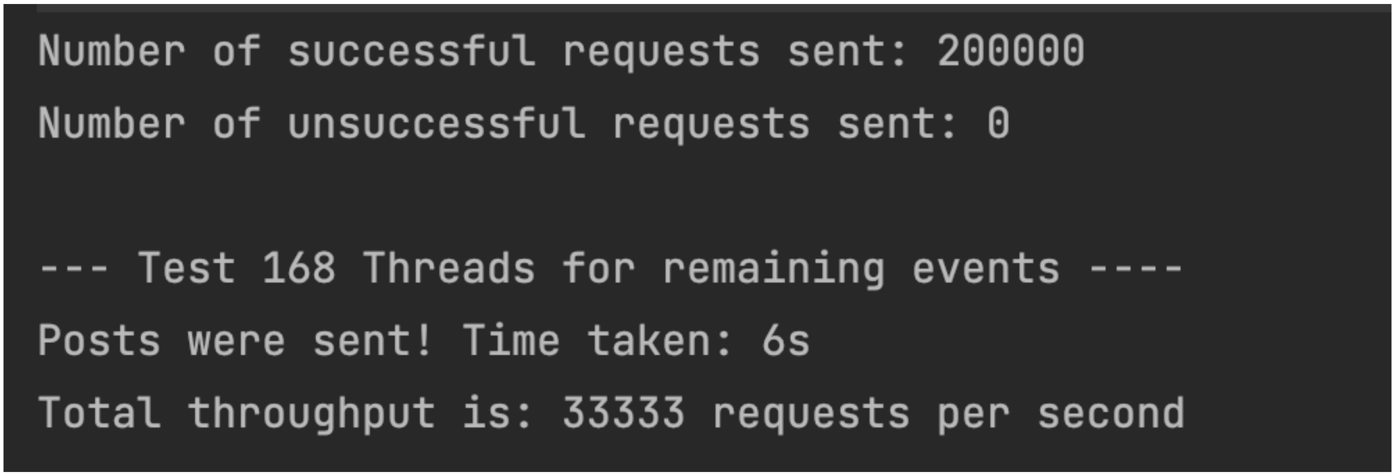
```
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 100 Threads for remaining events ----
Posts were sent! Time taken: 6s
Total throughput is: 33333 requests per second
```

- **ec2: created 100 threads for remaining requests:**



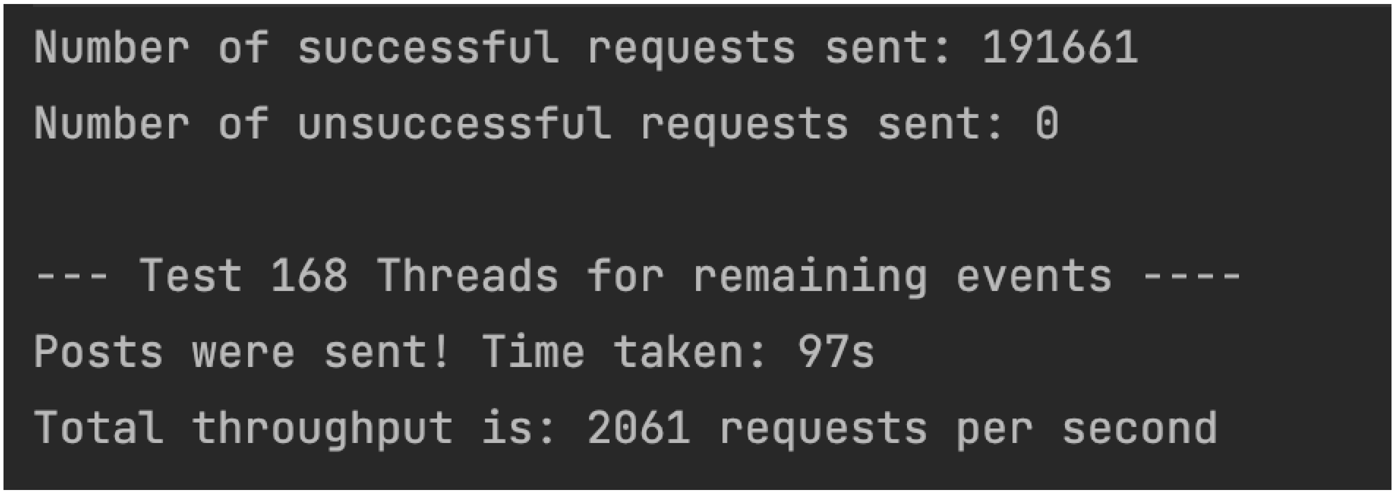
```
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 100 Threads for remaining events ----
Posts were sent! Time taken: 130s
Total throughput is: 1538 requests per second
```

- **local: created 168 threads for remaining requests:**



```
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 168 Threads for remaining events ----
Posts were sent! Time taken: 6s
Total throughput is: 33333 requests per second
```

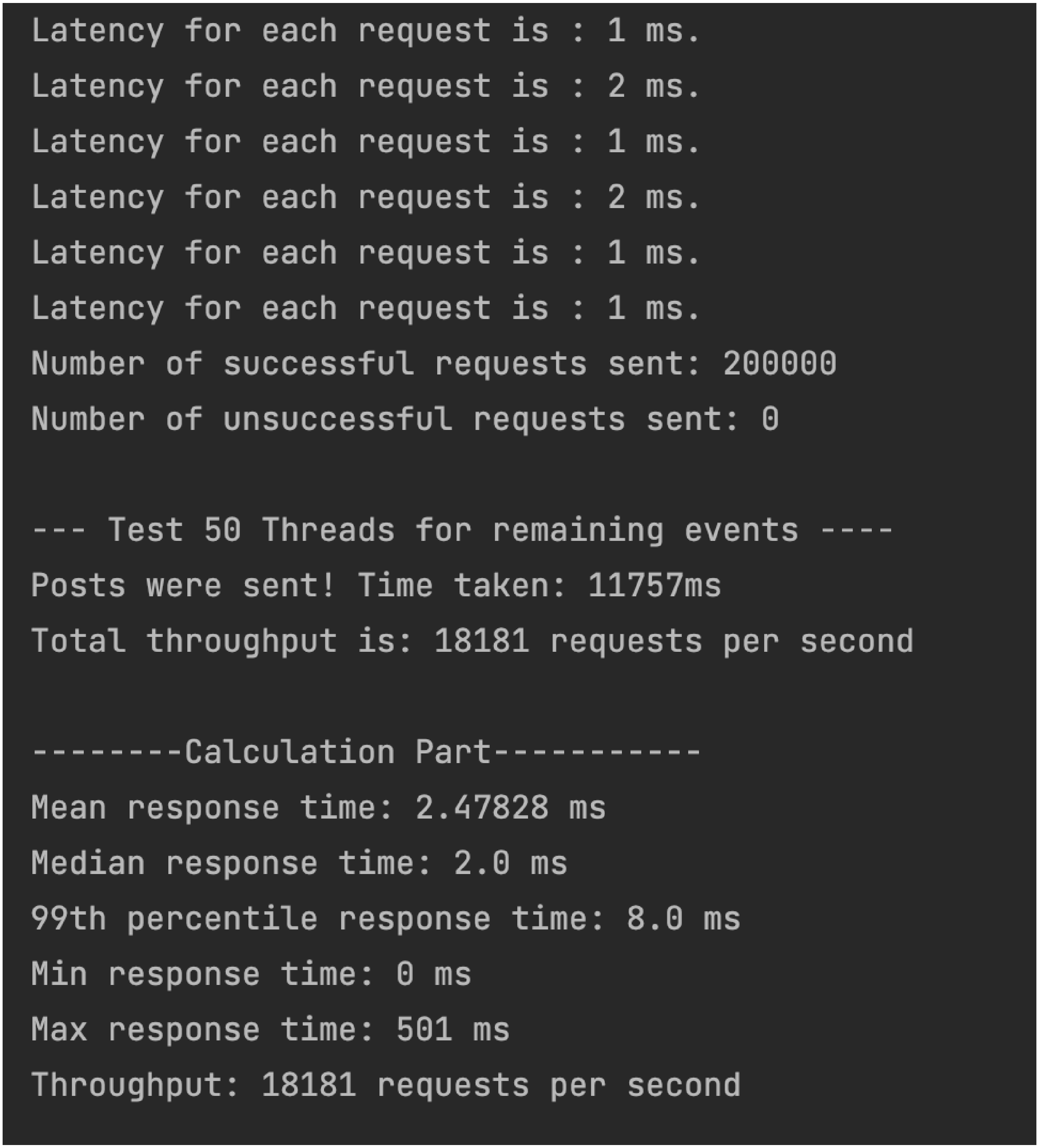
- **ec2: created 168 threads for remaining requests:**



```
Number of successful requests sent: 191661
Number of unsuccessful requests sent: 0

--- Test 168 Threads for remaining events ----
Posts were sent! Time taken: 97s
Total throughput is: 2061 requests per second
```

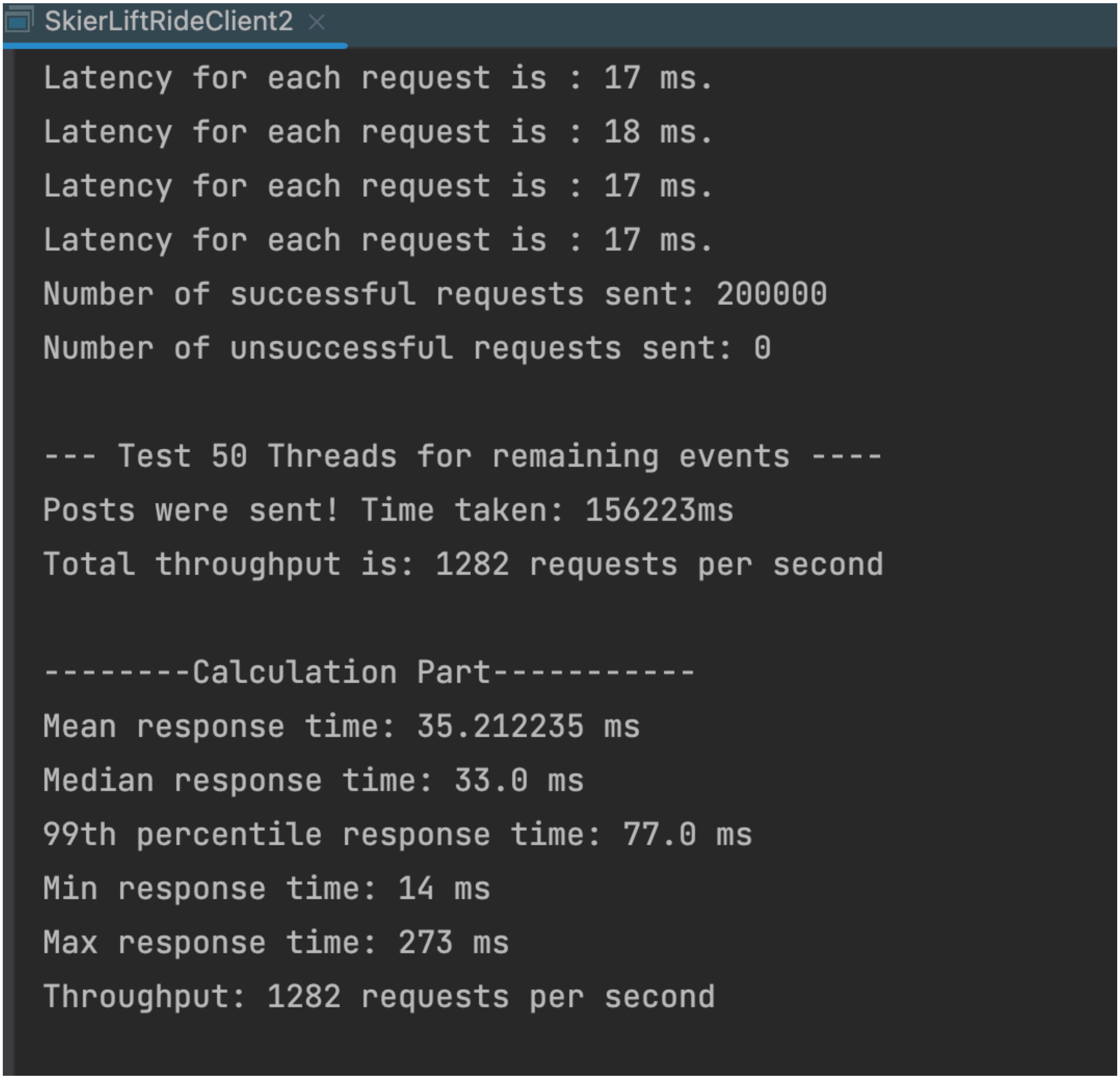
6. Actual throughput -- Client Part 2

- **local: created 50 threads for remaining requests:**



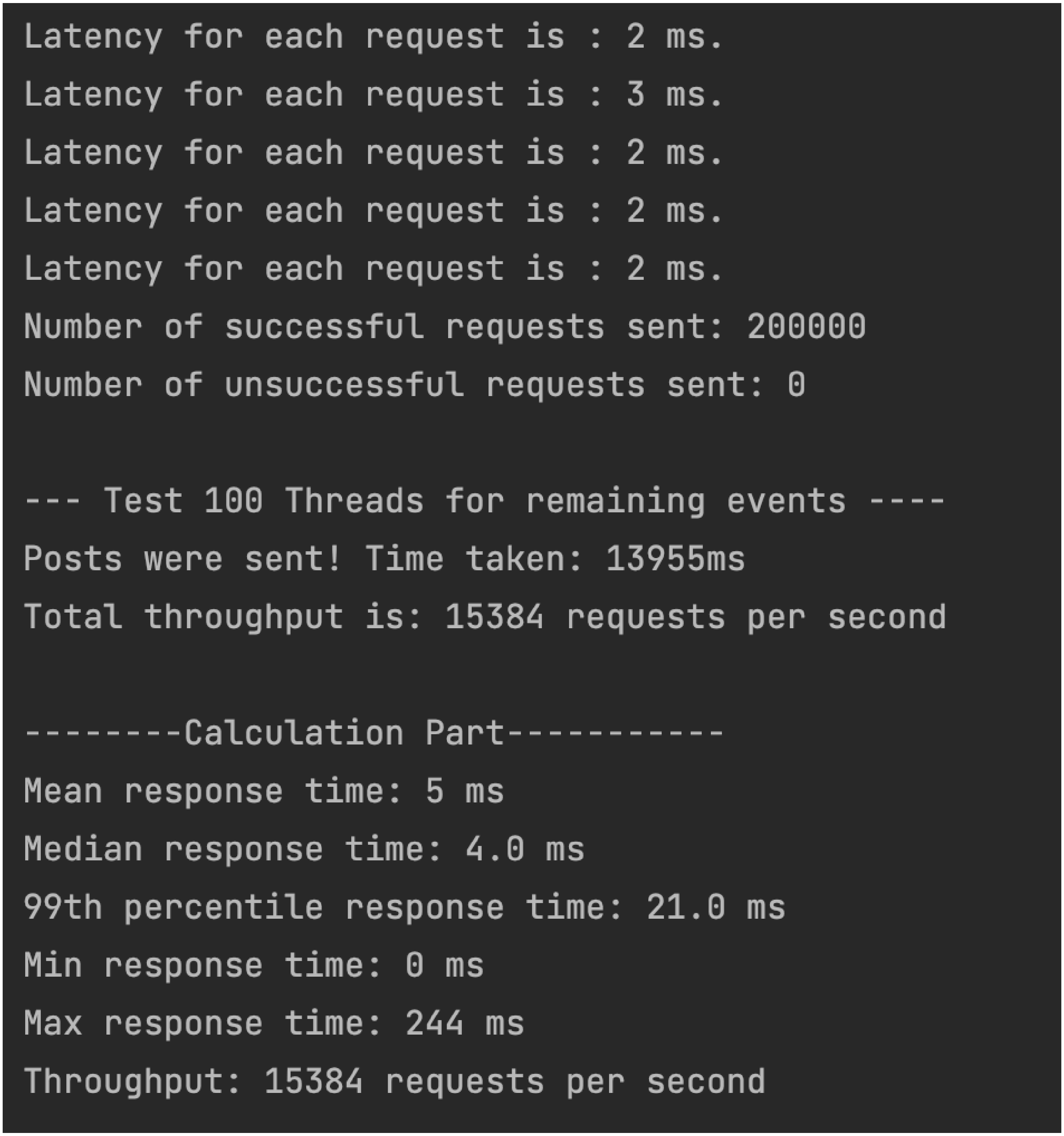
```
Latency for each request is : 1 ms.
Latency for each request is : 2 ms.
Latency for each request is : 1 ms.
Latency for each request is : 2 ms.
Latency for each request is : 1 ms.
Latency for each request is : 1 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 50 Threads for remaining events ----
Posts were sent! Time taken: 11757ms
Total throughput is: 18181 requests per second

--------Calculation Part-----------
Mean response time: 2.47828 ms
Median response time: 2.0 ms
99th percentile response time: 8.0 ms
Min response time: 0 ms
Max response time: 501 ms
Throughput: 18181 requests per second
```

- **ec2: created 50 threads for remaining requests:**



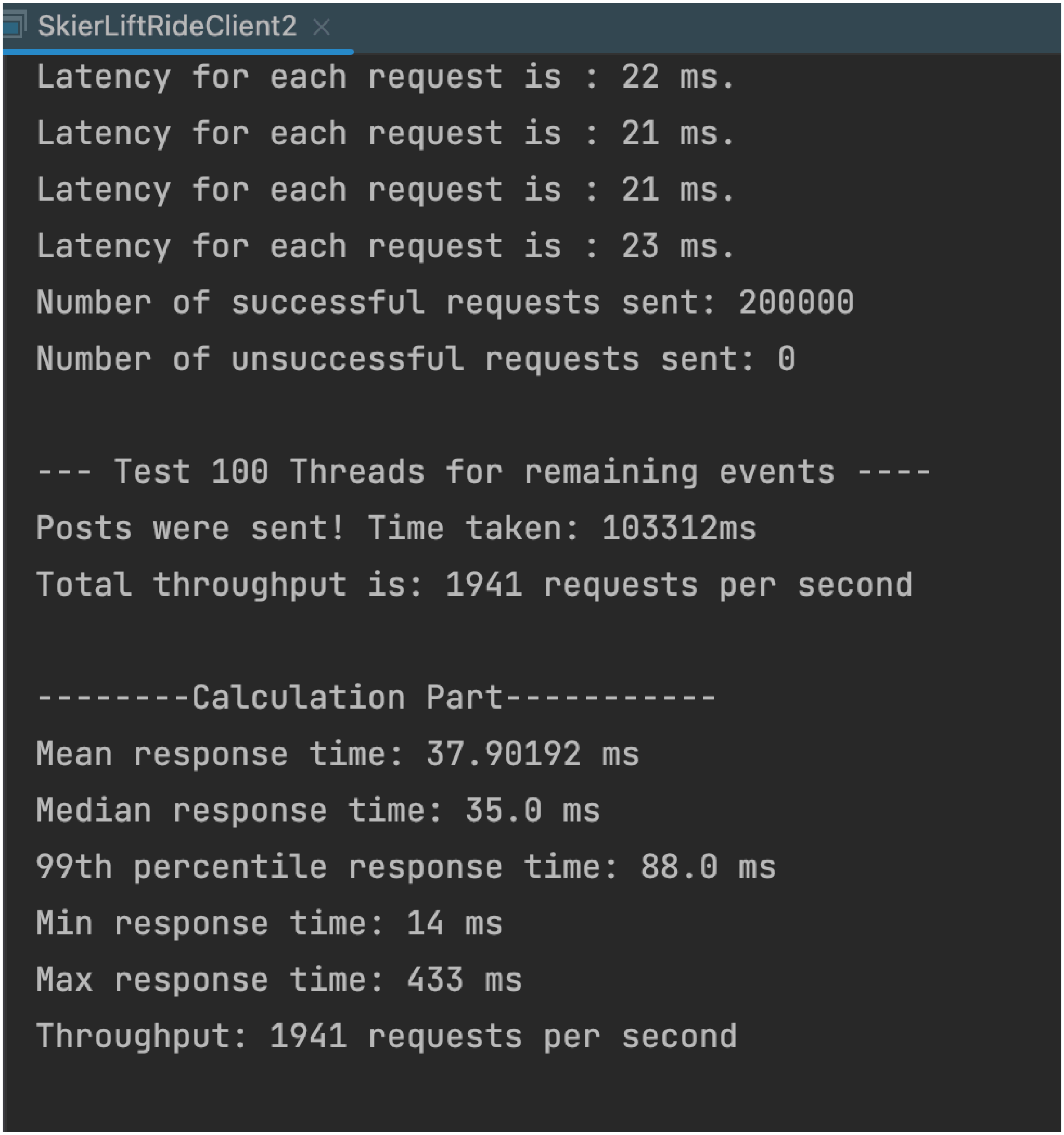
```
Latency for each request is : 17 ms.
Latency for each request is : 18 ms.
Latency for each request is : 17 ms.
Latency for each request is : 17 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 50 Threads for remaining events ----
Posts were sent! Time taken: 156223ms
Total throughput is: 1282 requests per second

--------Calculation Part-----------
Mean response time: 35.212235 ms
Median response time: 33.0 ms
99th percentile response time: 77.0 ms
Min response time: 14 ms
Max response time: 273 ms
Throughput: 1282 requests per second
```

- **local: created 100 threads for remaining requests:**



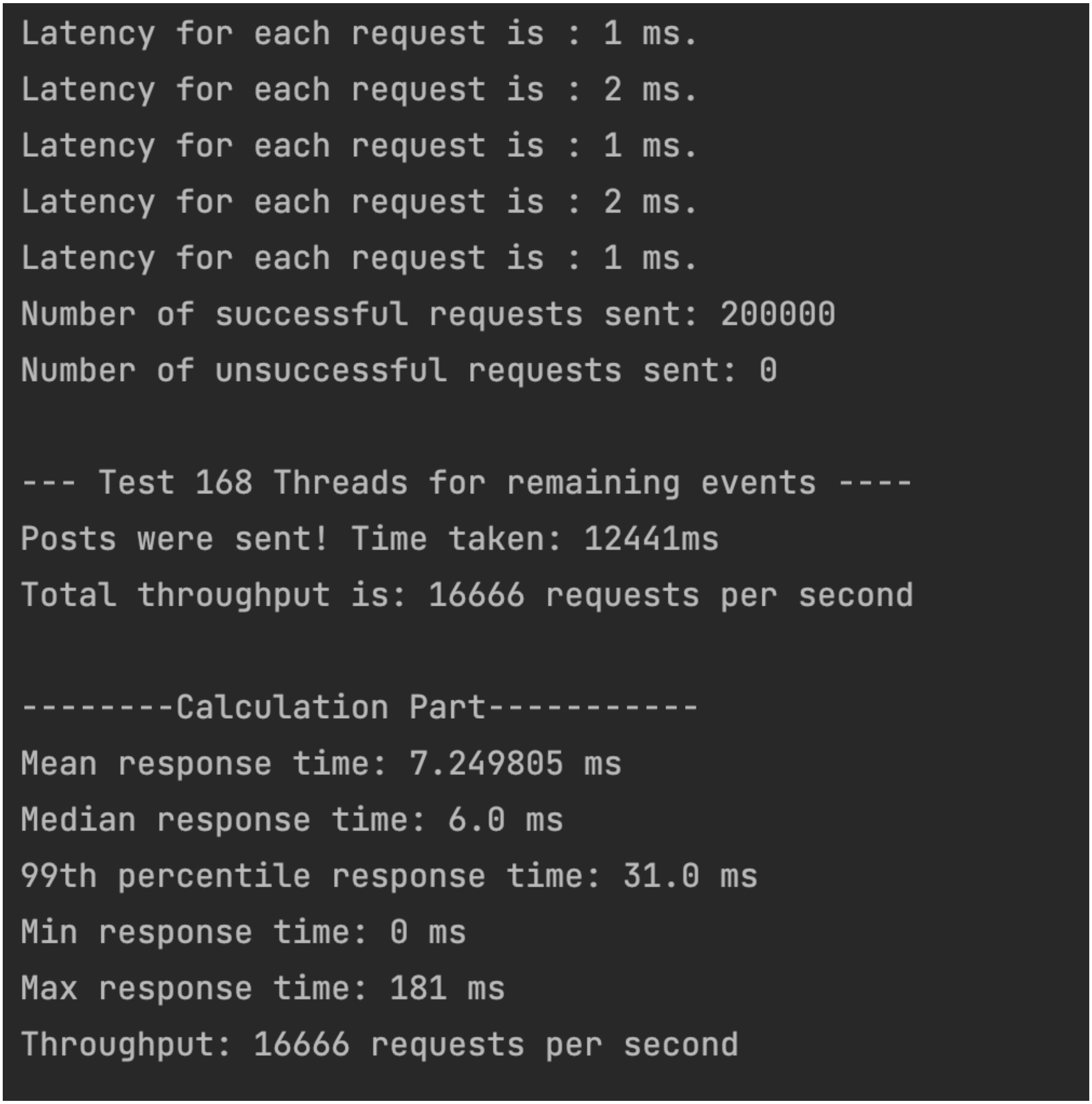
```
Latency for each request is : 2 ms.
Latency for each request is : 3 ms.
Latency for each request is : 2 ms.
Latency for each request is : 2 ms.
Latency for each request is : 2 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 100 Threads for remaining events ----
Posts were sent! Time taken: 13955ms
Total throughput is: 15384 requests per second

--------Calculation Part-----------
Mean response time: 5 ms
Median response time: 4.0 ms
99th percentile response time: 21.0 ms
Min response time: 0 ms
Max response time: 244 ms
Throughput: 15384 requests per second
```

- **ec2: created 100 threads for remaining requests:**



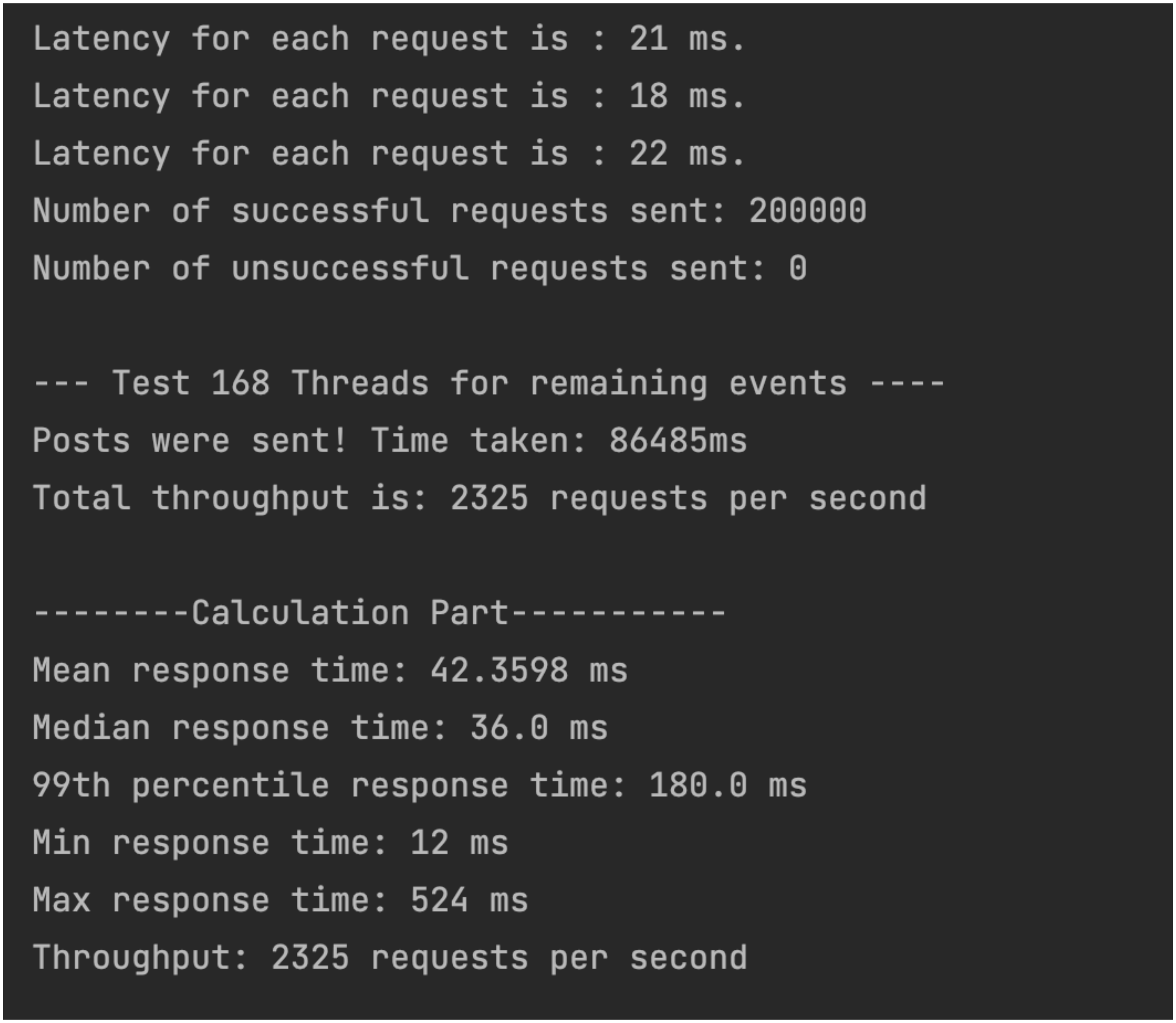
```
Latency for each request is : 22 ms.
Latency for each request is : 21 ms.
Latency for each request is : 21 ms.
Latency for each request is : 23 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 100 Threads for remaining events ----
Posts were sent! Time taken: 103312ms
Total throughput is: 1941 requests per second

--------Calculation Part-----------
Mean response time: 37.90192 ms
Median response time: 35.0 ms
99th percentile response time: 88.0 ms
Min response time: 14 ms
Max response time: 433 ms
Throughput: 1941 requests per second
```

- **local: created 168 threads for remaining requests:**



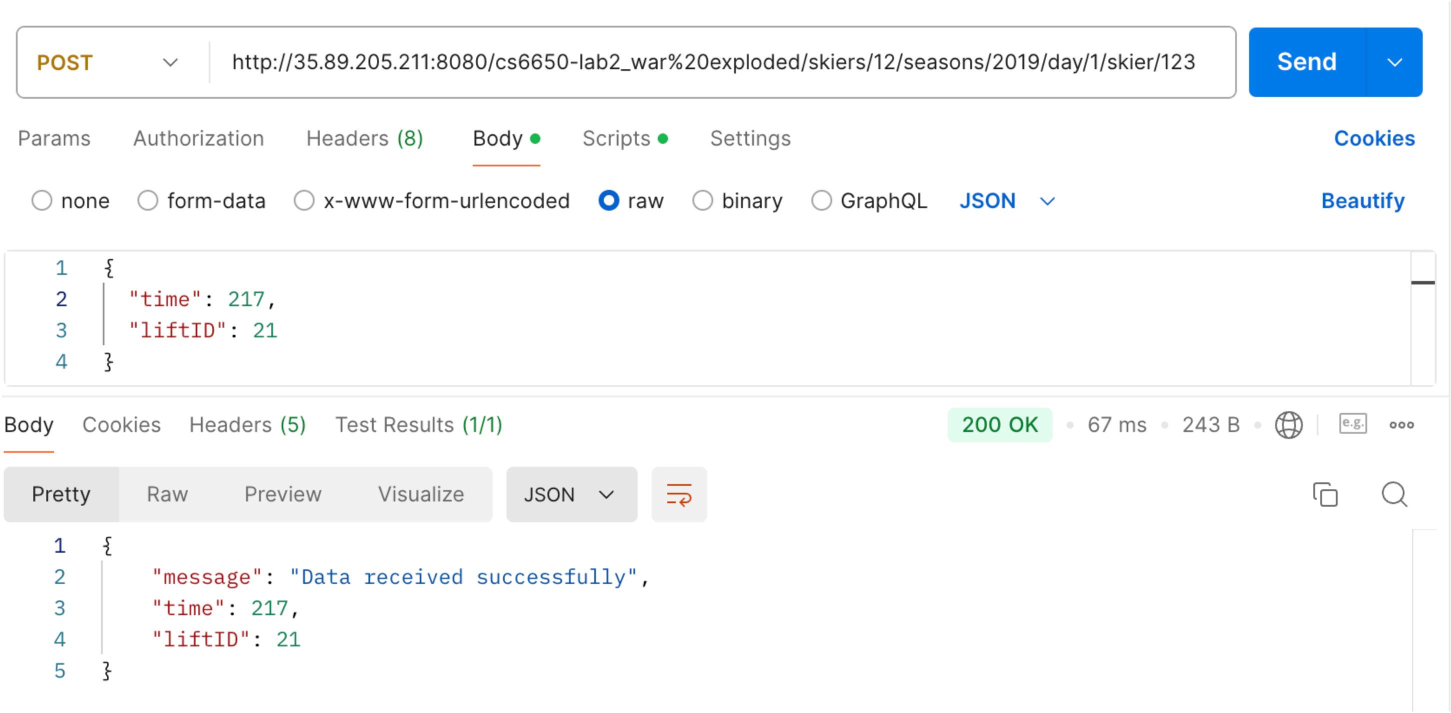
```
Latency for each request is : 1 ms.
Latency for each request is : 2 ms.
Latency for each request is : 1 ms.
Latency for each request is : 2 ms.
Latency for each request is : 1 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 168 Threads for remaining events ----
Posts were sent! Time taken: 12441ms
Total throughput is: 16666 requests per second

--------Calculation Part-----------
Mean response time: 7.249805 ms
Median response time: 6.0 ms
99th percentile response time: 31.0 ms
Min response time: 0 ms
Max response time: 181 ms
Throughput: 16666 requests per second
```

- **ec2: created 168 threads for remaining requests:**



```
Latency for each request is : 21 ms.
Latency for each request is : 18 ms.
Latency for each request is : 22 ms.
Number of successful requests sent: 200000
Number of unsuccessful requests sent: 0

--- Test 168 Threads for remaining events ----
Posts were sent! Time taken: 86485ms
Total throughput is: 2325 requests per second

--------Calculation Part-----------
Mean response time: 42.3598 ms
Median response time: 36.0 ms
99th percentile response time: 180.0 ms
Min response time: 12 ms
Max response time: 524 ms
Throughput: 2325 requests per second
```

7. EC2 screenshots

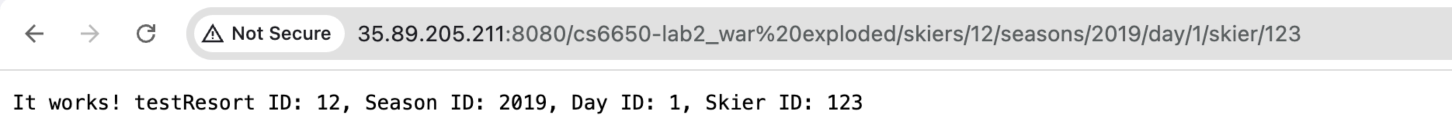


POST http://35.89.205.211:8080/cs6650-lab2_war%20exploded/skiers/12/seasons/2019/day/1/skier/123

```
{
  "time": 217,
  "liftID": 21
}
```

200 OK • 67 ms • 243 B

```
{
    "message": "Data received successfully",
    "time": 217,
    "liftID": 21
}
```



35.89.205.211:8080/cs6650-lab2_war%20exploded/skiers/12/seasons/2019/day/1/skier/123

```
It works! testResort ID: 12, Season ID: 2019, Day ID: 1, Skier ID: 123
```