Git URL: https://github.ccs.neu.edu/yihtian/CS6650/tree/master/Assignment2

Design:

On the server side, there are majorly 3 parts: servlets, data access layer, and database connector.

Connector part provides information of database and thus create new connection to database.

For database part, 2 schemas are designed. LiftRide schema is the database to record information such as skierId, dayId and etc., Stat schema is the database to record respond time for each request. SkierDAO provide 2 functions, first allow new data from a request to be inserted into database as a query, the second allows client to get vertical information from existed data by providing a skierId and a dayId. StatsDAO also provides 2 functions that both allow client to write in respond time information and check the mean and max response time at runtime.

For servlets, SkierServlet doPost method accepts requests from client which sends the information of a lift ride, then insert info from request into database LiftRide through SkierDao then send back response. After sending back response, record the total response to process 1 request and insert that information into Stat database by using StatDao. SkierServlet doGet method accepts skierId and dayId from request and using these 2 things to lookup vertical information in existed LiftRide data by calling the get function in SkierDao.
For StatServlet, only doGet method is designed, when client sends a request to get information, the servlet look through Stat table and get the mean and max for current data and send back this information as a APIStats.

On the client side, the design is similar to assignment1. In a single thread, the thread will send required number of post requests, but at this time, if the thread is running in the third phase will be checked, if true then the thread will send out additional get request. Third phase information will be an input as Boolean. Then each phase creaetes required number of threads, and all phases will be executed in main function.

Results:
For single server:

```
=============Progress Finished=============
Total threads: 32
Total requests posted: 440000
Execution time: 1449774 millisecs
Total successful request sent: 440000

=============Statistic Results=============
Mean response time: 69 millisecs
Median response time: 64 millisecs
99th percentile: 180 millisecs
Max response time: 1257 millisecs
Throughput: 303

Process finished with exit code 0
```

```
============Progress Finished============
Total threads: 64
Total requests posted: 440000
Execution time: 818521 millisecs
Total successful request sent: 439488

============Statistic Results============
Mean response time: 92 millisecs
Median response time: 86 millisecs
99th percentile: 238 millisecs
Max response time: 805 millisecs
Throughput: 537

Process finished with exit code 0
```

```
=============Progress Finished=============
Total threads: 128
Total requests posted: 440000
Execution time: 911002 millisecs
Total successful request sent: 439488

=============Statistic Results=============
Mean response time: 206 millisecs
Median response time: 210 millisecs
99th percentile: 484 millisecs
Max response time: 5554 millisecs
Throughput: 482

Process finished with exit code 0
```

```
=============Progress Finished=============
Total threads: 256
Total requests posted: 440000
Execution time: 794592 millisecs
Total successful request sent: 439296

=============Statistic Results=============
Mean response time: 364 millisecs
Median response time: 345 millisecs
99th percentile: 1535 millisecs
Max response time: 4094 millisecs
Throughput: 553

Process finished with exit code 0
```

For load balanced server:

```
1574850304389,GET,20,200
============Progress Finished============
Total threads: 32
Total requests posted: 440000
Execution time: 599905 millisecs
Total successful request sent: 439975

============Statistic Results============
Mean response time: 32 millisecs
Median response time: 25 millisecs
99th percentile: 183 millisecs
Max response time: 8758 millisecs
Throughput: 734

Process finished with exit code 0
```

```
=============Progress Finished=============
Total threads: 64
Total requests posted: 440000
Execution time: 1333294 millisecs
Total successful request sent: 439488

=============Statistic Results=============
Mean response time: 155 millisecs
Median response time: 43 millisecs
99th percentile: 912 millisecs
Max response time: 2353 millisecs
Throughput: 329

Process finished with exit code 0
```

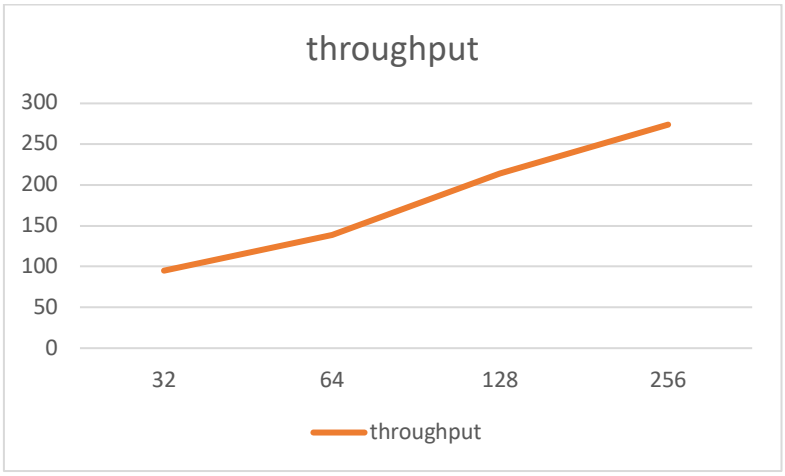
```
=============Progress Finished=============
Total threads: 128
Total requests posted: 440000
Execution time: 252889 millisecs
Total successful request sent: 439478

=============Statistic Results=============
Mean response time: 51 millisecs
Median response time: 36 millisecs
99th percentile: 209 millisecs
Max response time: 8217 millisecs
Throughput: 1743

Process finished with exit code 0
```

```
=============Progress Finished=============
Total threads: 256
Total requests posted: 440000
Execution time: 216842 millisecs
Total successful request sent: 439290

=============Statistic Results=============
Mean response time: 97 millisecs
Median response time: 40 millisecs
99th percentile: 545 millisecs
Max response time: 9397 millisecs
Throughput: 2033

Process finished with exit code 0
```

Runtime Statistics:
Called get for 3 different times during run time, and got following information.

```
r_war_exploded/statistics/"
{"endpointStats":[{"URL":"/","operation":"GET","mean":92,"max":99}]}(
ibandeMacBook-Pro:~ y_tian$
r_war_exploded/statistics/
{"endpointStats":[{"URL":"/","operation":"GET","mean":92,"max":99}]}
ibandeMacBook-Pro:~ y_tian$
{"endpointStats":[{"URL":"/","operation":"GET","mean":89,"max":99}]}(
ibandeMacBook-Pro:~ y_tian$
```

Charts for single server:



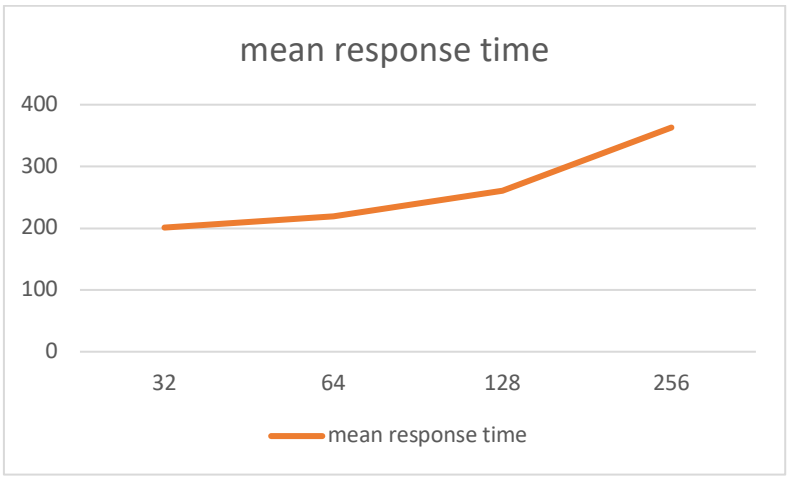


Chart for load balanced server: