A2 Server Design

Chenxi Cai

Github url: https://github.com/xiaohaiguicc/DistributedSystem/tree/master/assignment2

1.Server:

Based on A1, we need to add more options in 'doPost' and 'doGet'.

1) Create LiftRide class
   Every time a post request is made, a LiiftRide object should be created and all the information (skierId, resortId, seasonId, dayId, time, liftId) will be inset into database.
2) As for 'doPost', we need to parse the urlPath and body to get time and liftId information. Also, we need to split urlPath to get other information, and then create LiftRide object.
3) As for 'doGet', we need to split urlPath to get information and then select from database to return corresponding object.
4) Create new StatisticServlet. This is for the 'Collect Runtime Statistics' part. We need to build a new servlet to get statistics.

2. DAO:

Based on lab6, in order to connect Tomcat servlet to MySQL, we need to us JDBC.

1) The DBCPDataSource class is to set up DB connection manager. Because of connection error, I chose to create a getConnection() function and closeConnection() function to start and close the connection.
2) LiftRide class: This is to build the LiftRide object with necessary information.
3) LiftRideDao class: This class is to create a DAO layer between my servlets and database. It deals with all operations in database, like inserting or selecting.
   I. The 'createLiftRide()' function is to insert LiftRide object into database. Here I use a insert command to insert data.
   II. Because insert data one by one is too costly, I choose to use a BlockingQueue to store the insert statement. Once the statement in the queue is more than n, all the statements in queue will be executed with batch(batchInsert()), all the data will be inserted into table 'LiftRide'. Also, I use a timer with period 1s, which means that, every second, if the queue is not empty, the statements in queue will also be exeuted.
   III. The 'createRecord()' function is the same as 'createLiftRide()' function, in order to insert statistics into database.
   IV. The 'getVerticalForSpecificDay()' is to deal with the skiers get request with skierID, resortID, dayID and seasonID. Here I use SELECT command in database to build the query and get data from database.
   V. The 'getVerticalForSpecificResort ()' are same as g'etVerticalForSpecificDay()' for the other get request.
   VI. The 'getStatistics()' is for statistcs get request. Get and calculate the mean and max latencies for endpoint that store in the table 'statistcs'.
4) Statistics class: This is to build the class for table 'statistics' with information url,

requestType, latency for mean calculation and latency for max calculation.

3. Client:

Only modified phase 3. After making a post request, also make a get request.

4. Table: Build LiftRide database

1) Create a 'LiftRide' table, with primary key is an auto increasement id. Other columns are 'skierId', 'resortId', 'seasonId', 'dayId', 'time', 'liftId'. Insert data with every skier post request. Get data with every skier get request.
2) Create a 'statistcs' table, with primary key is an auto increasement id. Other columns are 'url', 'requestType', 'mean', 'max'. Insert data with every skier post/get request. Get data with every Statitcs get request.

Single Server Test

```
Total number of requests sent: 358700
Total number of successful responses: 358700
Total number of failed responses: 0
Wall time: 546574 milliseconds
The mean of all latencies: 81.0 milliseconds.
The median of all latencies: 90.0 milliseconds.
The throughput: 0.65626978233755 milliseconds.
The 99th percentile of latencies: 329.0 milliseconds.
The max response time: 1366.0 milliseconds.

Process finished with exit code 0
```

32 Threads

```
Total number of requests sent: 358284
Total number of successful responses: 358284
Total number of failed responses: 0
Wall time: 463195 milliseconds
The mean of all latencies: 115.0 milliseconds.
The median of all latencies: 99.0 milliseconds.
The throughput: 0.77350575896298 milliseconds.
The 99th percentile of latencies: 267.0 milliseconds.
The max response time: 2819.0 milliseconds.

Process finished with exit code 0
```

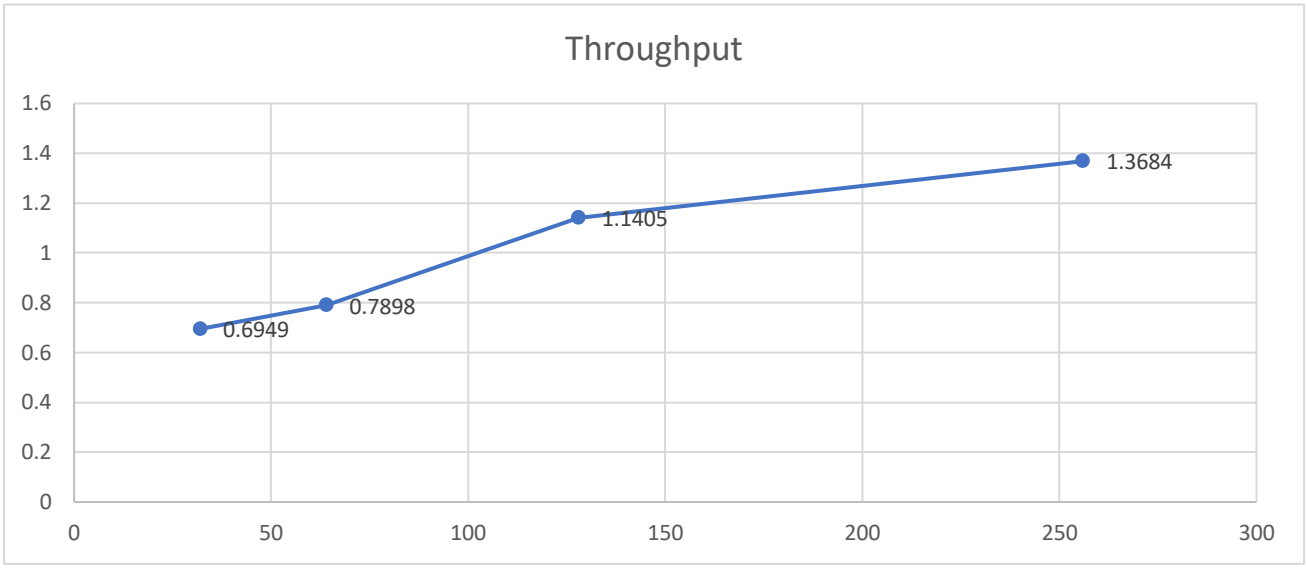
64 Threads

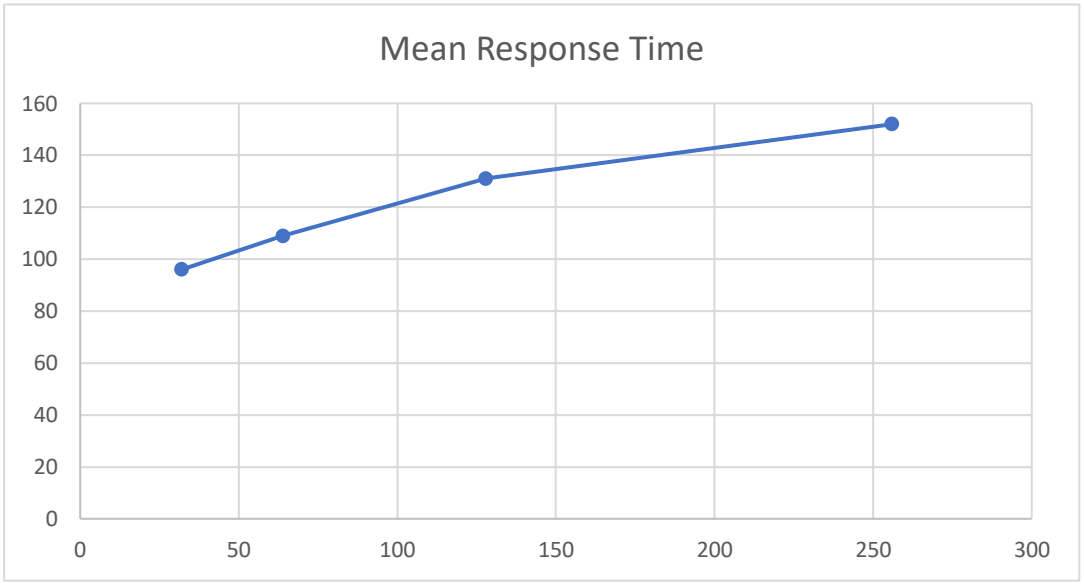
```
Total number of requests sent: 360032
Total number of successful responses: 360032
Total number of failed responses: 0
Wall time: 320986 milliseconds
The mean of all latencies: 135.0 milliseconds.
The median of all latencies: 121.0 milliseconds.
The throughput: 1.1216439346065555 milliseconds.
The 99th percentile of latencies: 386.0 milliseconds.
The max response time: 3540.0 milliseconds.
```

128 Threads

```
Total number of requests sent: 359680
Total number of successful responses: 359680
Total number of failed responses: 0
Wall time: 285948 milliseconds
The mean of all latencies: 180.0 milliseconds.
The median of all latencies: 6.0 milliseconds.
The throughput: 1.2578510778183445 milliseconds.
The 99th percentile of latencies: 2046.0 milliseconds.
The max response time: 5647.0 milliseconds.
```

256 Threads





Load Balanced Server Test

No much improvements.

```
Total number of requests sent: 356705
Total number of successful responses: 356705
Total number of failed responses: 0
Wall time: 513304 milliseconds
The mean of all latencies: 96.0 milliseconds.
The median of all latencies: 97.0 milliseconds.
The throughput: 0.69491957988314 milliseconds.
The 99th percentile of latencies: 589.0 milliseconds.
The max response time: 1820.0 milliseconds.

Process finished with exit code 0
```

32 Threads

```
Total number of requests sent: 359527
Total number of successful responses: 359527
Total number of failed responses: 0
Wall time: 455213 milliseconds
The mean of all latencies: 109.0 milliseconds.
The median of all latencies: 97.0 milliseconds.
The throughput: 0.78979950057149 milliseconds.
The 99th percentile of latencies: 287.0 milliseconds.
The max response time: 2781.0 milliseconds.

Process finished with exit code 0
```

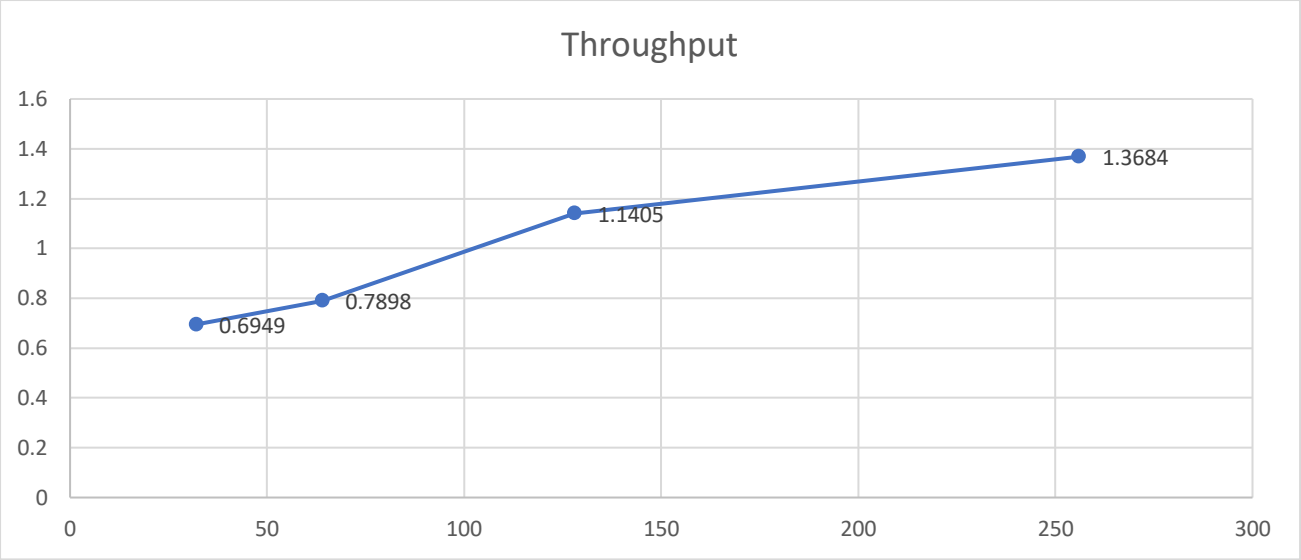
64 Threads

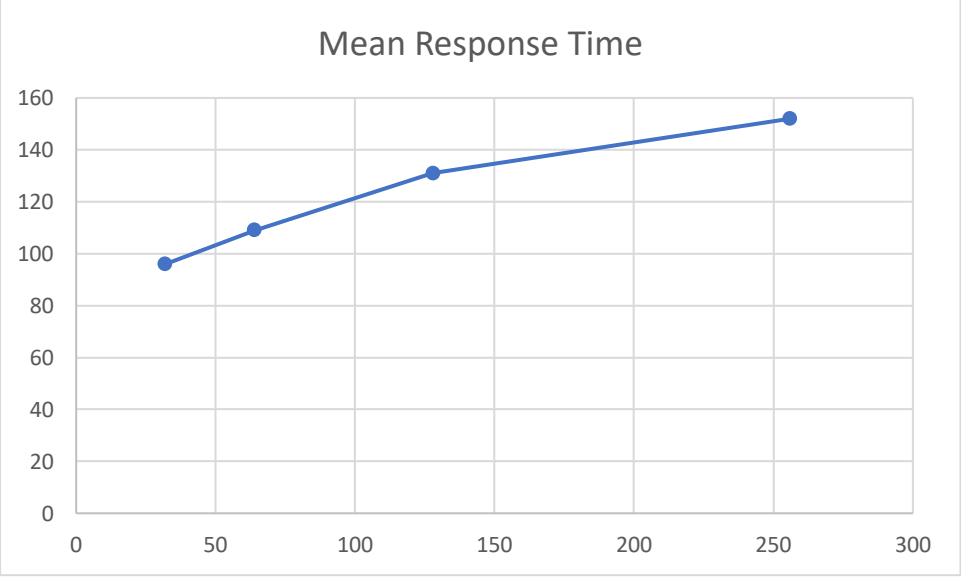
```
Total number of requests sent: 356916
Total number of successful responses: 356916
Total number of failed responses: 0
Wall time: 312949 milliseconds
The mean of all latencies: 141.0 milliseconds.
The median of all latencies: 129.0 milliseconds.
The throughput: 1.14049254034762 milliseconds.
The 99th percentile of latencies: 391.0 milliseconds.
The max response time: 3890.0 milliseconds.

Process finished with exit code 0
```

128 Threads

```
Total number of requests sent: 359616
Total number of successful responses: 359616
Total number of failed responses: 0
Wall time: 262792 milliseconds
The mean of all latencies: 192.0 milliseconds.
The median of all latencies: 151.0 milliseconds.
The throughput: 1.368443483819903 milliseconds.
The 99th percentile of latencies: 768.0 milliseconds.
The max response time: 3874.0 milliseconds.

Process finished with exit code 0
```

256 Threads





Runtime Statistics Collection

```
Total number of requests sent: 359676
Total number of successful responses: 359676
Total number of failed responses: 4
Wall time: 370556 milliseconds
The mean of all latencies: 125.0 milliseconds.
The median of all latencies: 5.0 milliseconds.
The throughput: 0.9706387158756031 milliseconds.
The 99th percentile of latencies: 1284.0 milliseconds.
The max response time: 3329.0 milliseconds.
```


```
[{"url":"resortID/seasonID/dayID/skierID","requestType":"
GET","mean":125,"max":3329},{"url":"resortID/seasonIDchch
```


We can see that the wall time is longer and some requests failed.