Assignment 2 write up

Github: https://github.com/AngelaJiang1225/CS6650

1. Design of database schema:

Here is my table structure and fields in Mysql:

a. Table `lifts_vertical`:

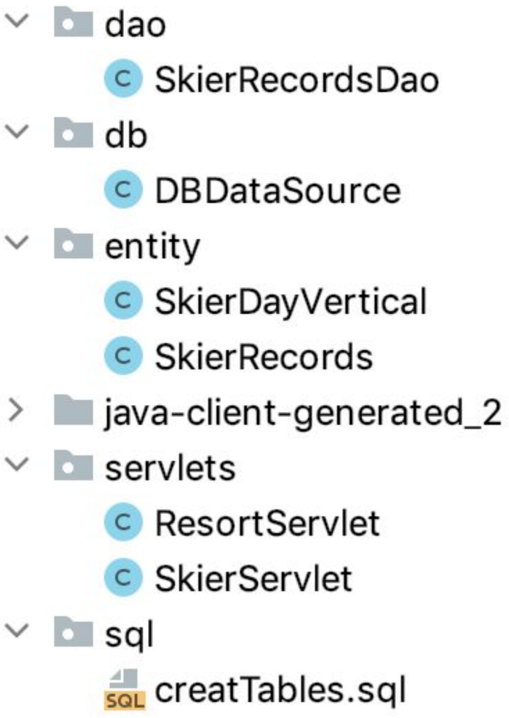
```
lift_id          int not null AUTO_INCREMENT primary key,
vertical         int  not null
```

b. Table `skier_records`:

```
id             int   not null AUTO_INCREMENT primary key,
resort_id      int                          not null,
lift_id        int                          not null,
season_id      int                          not null,
day_id         int                          comment 'day number in the
season',
day_time       int    default 0 not null,
CONSTRAINT fk_skier_records_lift_id
   FOREIGN KEY (lift_id)
       REFERENCES lifts_vertical(lift_id)
```

2. Design of server:

a. Server structure:



i. db layer:

This layer is for creating connection with Mysql database. DBDataSource here is to interact with database skierlift and get access to tables inside.

ii. java-client-generated_2:

This package is imported from swagger, which provides models including class LiftRide, SkierVertical, TopTen and TopTenTopTenSkiers. I utilized them as return values of methods in Dao layer (the models provided by swagger is good for converting into json). Inside the package, SkiersApi, ApiResponse and ApiException are used in client layer.

iii. entity layer:

I created two entities, they are: SkierRecords and SkierDayVertical. This layer is treated as supplement of models in java-client-generated_2 to be used in dao layer as return value or input parameters for getSkierDayVertical and createSkierRecord methods.

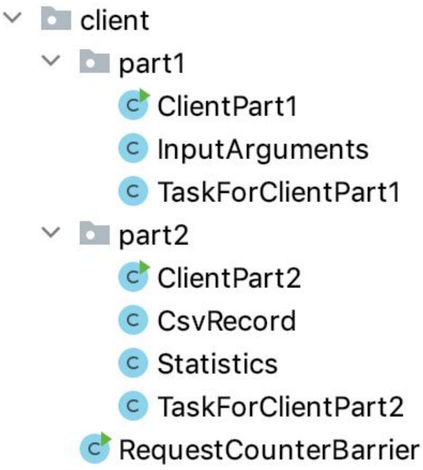
IV. dao layer:

This layer import classes in entity and java-client-generated_2 package, it is for implementing CRUD operations of skier_lift database, such as creating SkierRecord, getting skier day vertical, getting skier total vertical and getting top ten verticals.

V. Servlet layer:

This part is for accepting HTTP GET/POST request input and parsing url arguments. It imports classes in dao layer. If parsed valid requests, the class will call methods with requested path and parameters, else If invalid, return response code such as 400/404.

b. client structure



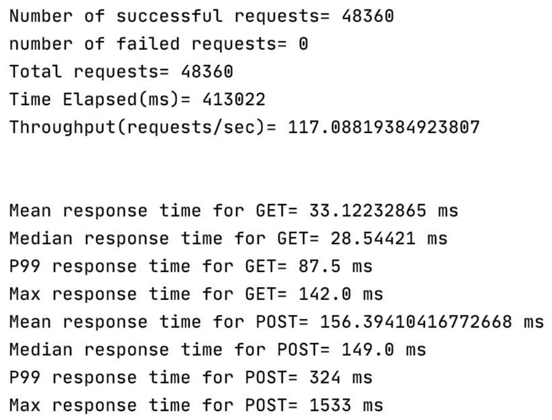
- client
  - part1
    - ClientPart1
    - InputArguments
    - TaskForClientPart1
  - part2
    - ClientPart2
    - CsvRecord
    - Statistics
    - TaskForClientPart2
  - RequestCounterBarrier

There are some changes in client part: post requests for each thread is increased from 100 to 1000. As assignment required, phase 3 threads have an extra 10 GETs for resort total in addition to 10 GETs for day totals. Dao layer is imported in client part, classes in swagger package such as ApiClient, ApiResponse are used in TaskForClientPart to write liftRide with http information.

3. Results

a. Single tests

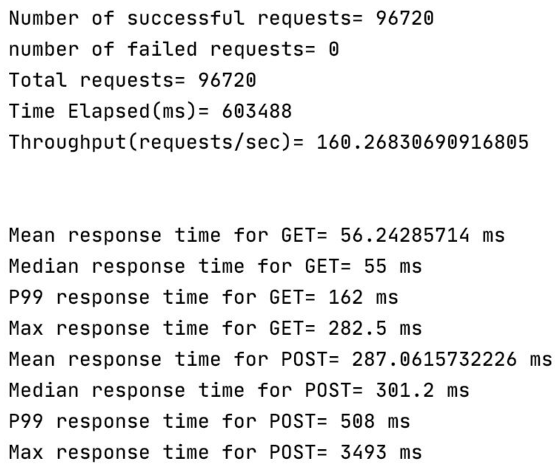
Thread 32:



```
Number of successful requests= 48360
number of failed requests= 0
Total requests= 48360
Time Elapsed(ms)= 413022
Throughput(requests/sec)= 117.08819384923807


Mean response time for GET= 33.12232865 ms
Median response time for GET= 28.54421 ms
P99 response time for GET= 87.5 ms
Max response time for GET= 142.0 ms
Mean response time for POST= 156.39410416772668 ms
Median response time for POST= 149.0 ms
P99 response time for POST= 324 ms
Max response time for POST= 1533 ms
```

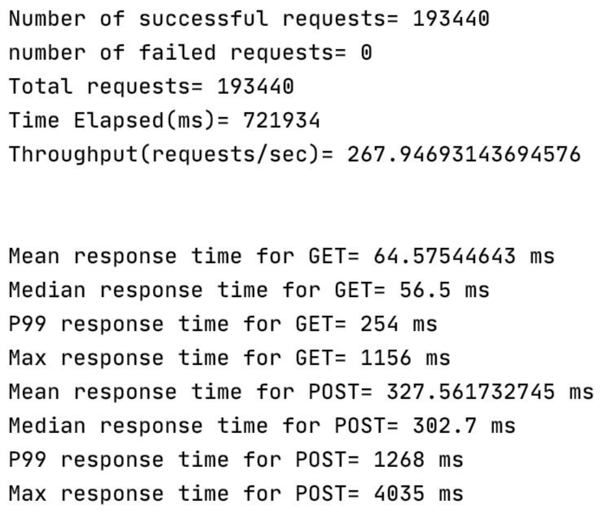
Thread 64:



```
Number of successful requests= 96720
number of failed requests= 0
Total requests= 96720
Time Elapsed(ms)= 603488
Throughput(requests/sec)= 160.26830690916805


Mean response time for GET= 56.24285714 ms
Median response time for GET= 55 ms
P99 response time for GET= 162 ms
Max response time for GET= 282.5 ms
Mean response time for POST= 287.0615732226 ms
Median response time for POST= 301.2 ms
P99 response time for POST= 508 ms
Max response time for POST= 3493 ms
```

Thread 128:



```
Number of successful requests= 193440
number of failed requests= 0
Total requests= 193440
Time Elapsed(ms)= 721934
Throughput(requests/sec)= 267.94693143694576


Mean response time for GET= 64.57544643 ms
Median response time for GET= 56.5 ms
P99 response time for GET= 254 ms
Max response time for GET= 1156 ms
Mean response time for POST= 327.561732745 ms
Median response time for POST= 302.7 ms
P99 response time for POST= 1268 ms
Max response time for POST= 4035 ms
```

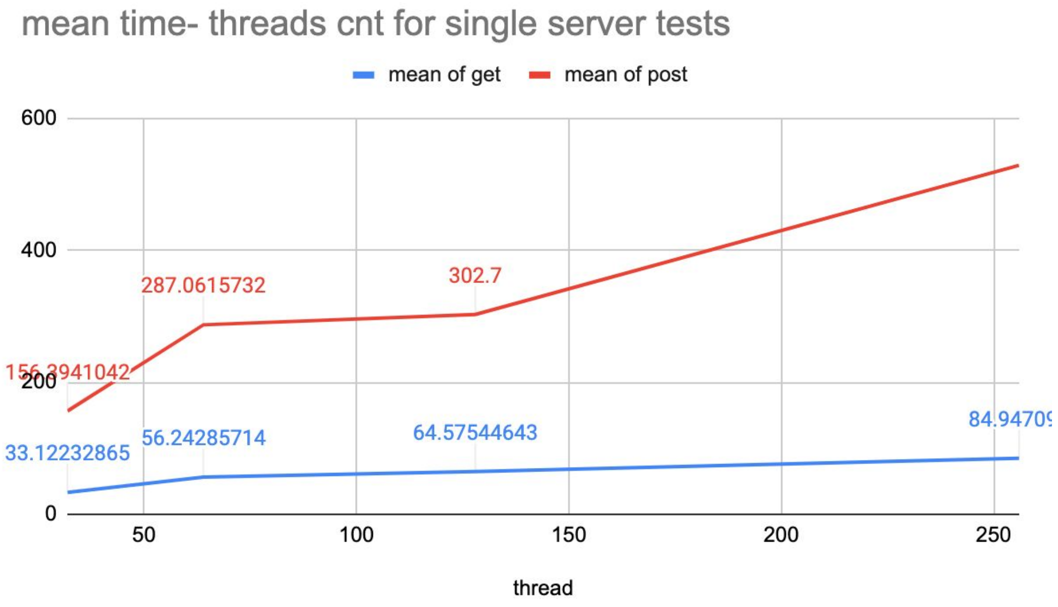
Thread 256

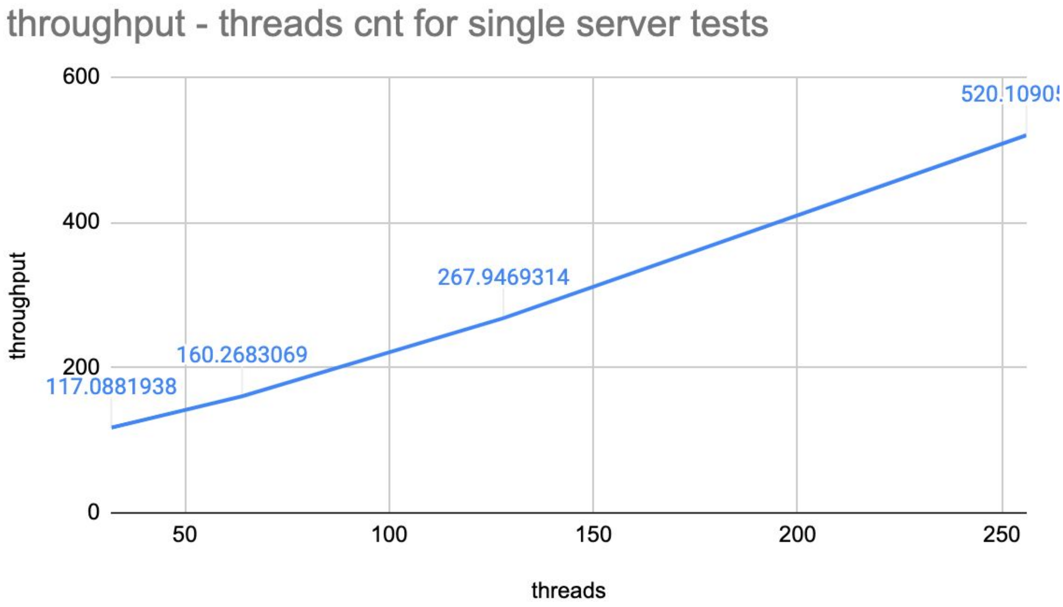
```
Number of successful requests= 386880
number of failed requests= 0
Total requests= 386880
Time Elapsed(ms)= 743844
Throughput(requests/sec)= 520.1090551244616


Mean response time for GET= 84.947098 ms
Median response time for GET= 73.5 ms
P99 response time for GET= 366.94709884309 ms
Max response time for GET= 2099.335 ms
Mean response time for POST= 528.632 ms
Median response time for POST= 407.0 ms
P99 response time for POST= 1338 ms
Max response time for POST= 4530 ms
```

charts for single server test:

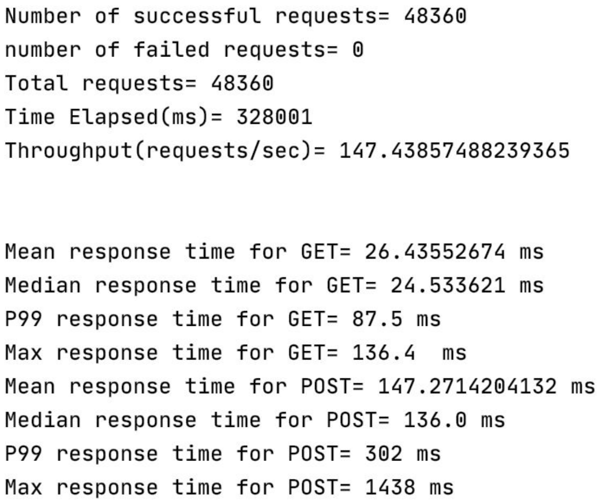
threads	mean of get	mean of post	throughput
32	33.12232865	156.3941042	117.0881938
64	56.24285714	287.0615732	160.2683069
128	64.57544643	302.7	267.9469314
256	84.947098	528.632	520.1090551





b. Load balance tests:

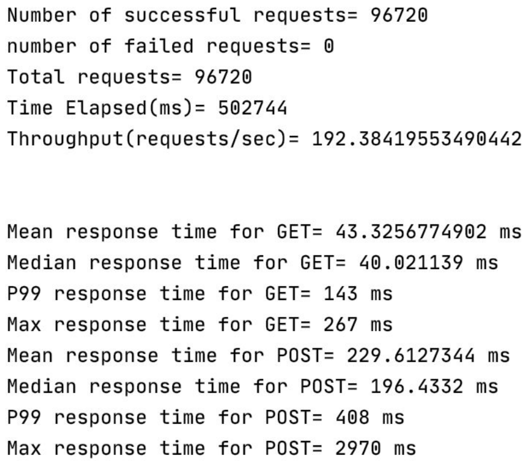
Thread 32:



```
Number of successful requests= 48360
number of failed requests= 0
Total requests= 48360
Time Elapsed(ms)= 328001
Throughput(requests/sec)= 147.43857488239365


Mean response time for GET= 26.43552674 ms
Median response time for GET= 24.533621 ms
P99 response time for GET= 87.5 ms
Max response time for GET= 136.4  ms
Mean response time for POST= 147.2714204132 ms
Median response time for POST= 136.0 ms
P99 response time for POST= 302 ms
Max response time for POST= 1438 ms
```

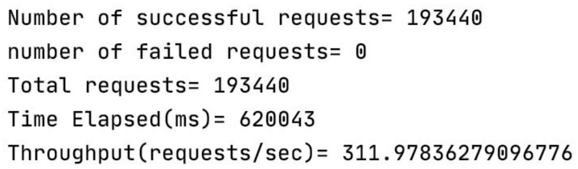
Thread 64:



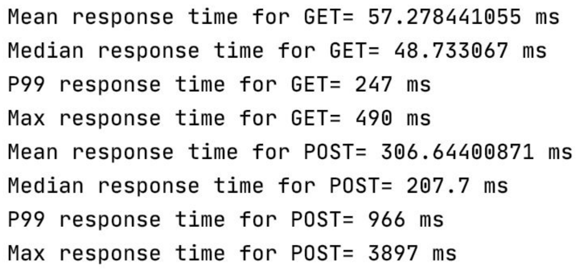
```
Number of successful requests= 96720
number of failed requests= 0
Total requests= 96720
Time Elapsed(ms)= 502744
Throughput(requests/sec)= 192.38419553490442


Mean response time for GET= 43.3256774902 ms
Median response time for GET= 40.021139 ms
P99 response time for GET= 143 ms
Max response time for GET= 267 ms
Mean response time for POST= 229.6127344 ms
Median response time for POST= 196.4332 ms
P99 response time for POST= 408 ms
Max response time for POST= 2970 ms
```

Thread 128:

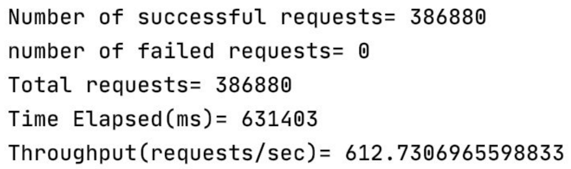


```
Number of successful requests= 193440
number of failed requests= 0
Total requests= 193440
Time Elapsed(ms)= 620043
Throughput(requests/sec)= 311.97836279096776
```

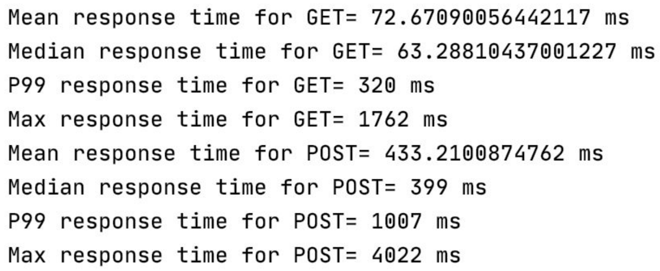


```
Mean response time for GET= 57.278441055 ms
Median response time for GET= 48.733067 ms
P99 response time for GET= 247 ms
Max response time for GET= 490 ms
Mean response time for POST= 306.64400871 ms
Median response time for POST= 207.7 ms
P99 response time for POST= 966 ms
Max response time for POST= 3897 ms
```

Thread 256:



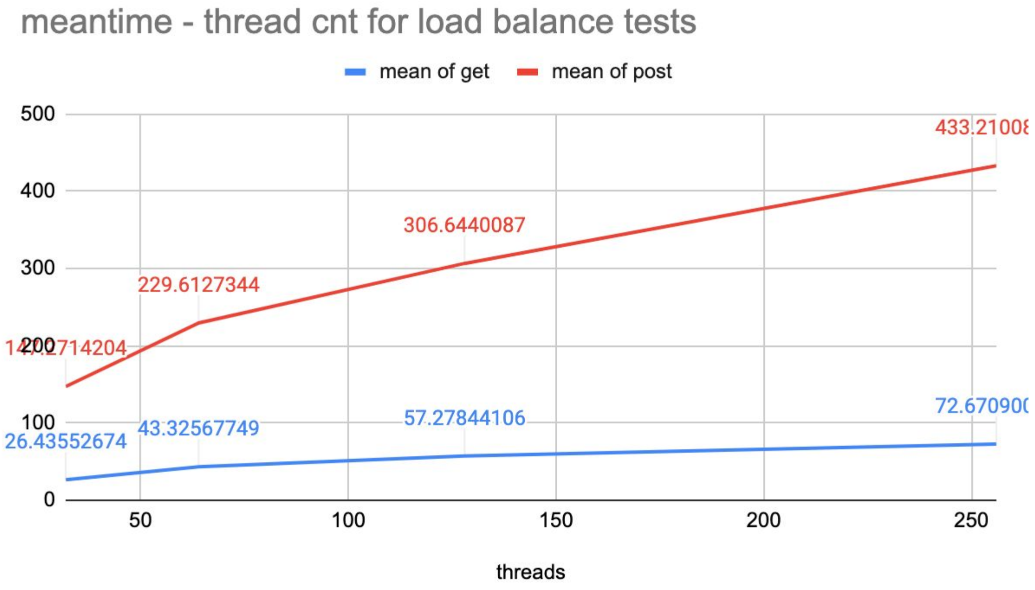
```
Number of successful requests= 386880
number of failed requests= 0
Total requests= 386880
Time Elapsed(ms)= 631403
Throughput(requests/sec)= 612.7306965598833
```

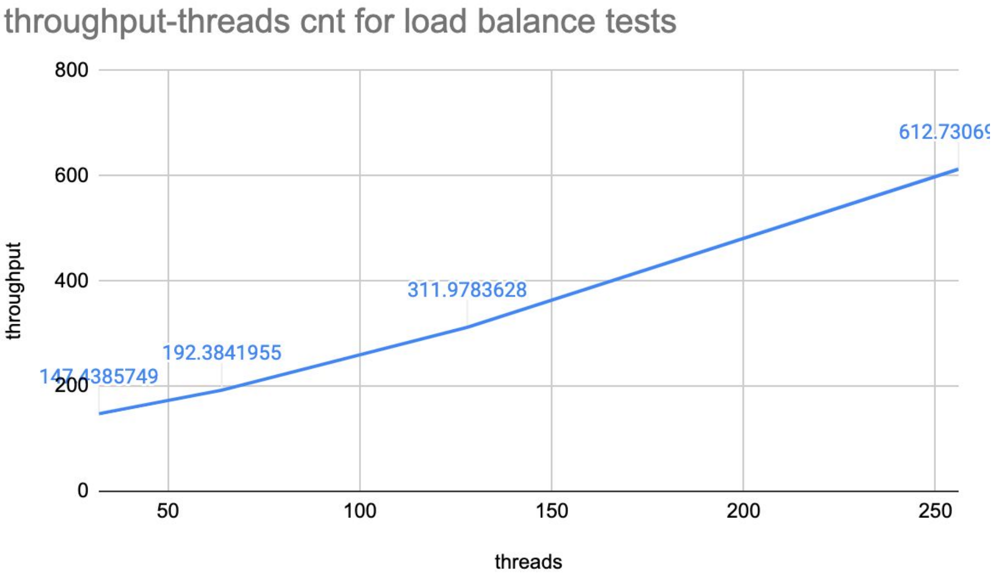


```
Mean response time for GET= 72.67090056442117 ms
Median response time for GET= 63.28810437001227 ms
P99 response time for GET= 320 ms
Max response time for GET= 1762 ms
Mean response time for POST= 433.2100874762 ms
Median response time for POST= 399 ms
P99 response time for POST= 1007 ms
Max response time for POST= 4022 ms
```

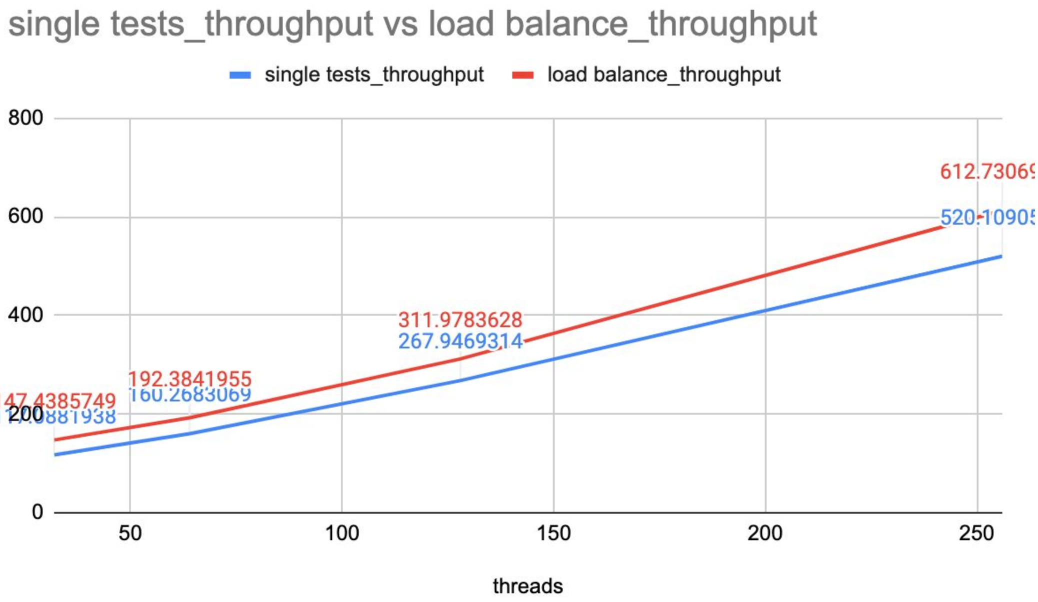
charts for load balance:

threads	mean of get	mean of post	throughput
32	26.43552674	147.2714204	147.4385749
64	43.32567749	229.6127344	192.3841955
128	57.27844106	306.6440087	311.9783628
256	72.67090056	433.2100875	612.7306966


meantime - thread cnt for load balance tests
mean of get
mean of post
500
400
300
200
100
0
433.21008
306.6440087
229.6127344
147.2714204
72.670900
57.27844106
43.32567749
26.43552674
50
100
150
200
250
threads


throughput-threads cnt for load balance tests
800
600
400
200
0
throughput
612.73069
311.9783628
192.3841955
147.4385749
50
100
150
200
250
threads

c. comparison of single tests and load balance tests:



From the plots above, we can see that load balance has better performance than single server when running different threads.