Client Design

Chenxi Cai

Guthub url:
https://github.com/xiaohaiguicc/DistributedSystem

The whole packages have two parts, Server and Client.

1. Server

The server uses lab1's code and have two methods 'doGet' and 'doPost'. These two methods will receive request and check if the url and body (for post) is valid, then return corresponding response.
According Swagger API, there will be two servlets, Skier and Resorts, so I wrote two similar servlets. The differences come from request parameter format and response body.

2. Client

Client has two part according to assignment 1

(1) Part1 have two parts, client and thread.

The thread class is to build a single thread and assign task to it. Every thread needs to set ip address and port, using swagger api to get the response. The api will call the servlets and return response.

The client class deal with three phases. I use countdown to count if 10% of threads finish executing. Every time, when first countdown is 0, the phase can begin and decrease second count down. Then the second countdown will be the first count down for the next phase. This logic will help to control three phases execute in order.

(2) Part2 is to calculate records and data

The record class is to record every POST request and response.

The Data calculate the mean, median, throughput, max, p99 response time…
and generate csv file.

256

```
Total number of requests sent: 359552
Total number of successful responses: 359552
Total number of failed responses: 0
Wall time is: 164298 milliseconds
----------------------------------------
The mean of all latencies: 87 milliseconds.
The median of all latencies: 81 milliseconds.
The throughput: 2 milliseconds.
The 99th percentile of latencies: 178 milliseconds.
The max response time: 1207 milliseconds.

Process finished with exit code 0
```

128

```
Total number of requests sent: 359552
Total number of successful responses: 359552
Total number of failed responses: 0
Wall time is: 321057 milliseconds
----------------------------------------
The mean of all latencies: 86 milliseconds.
The median of all latencies: 79 milliseconds.
The throughput: 1 milliseconds.
The 99th percentile of latencies: 184 milliseconds.
The max response time: 1215 milliseconds.

Process finished with exit code 0
```

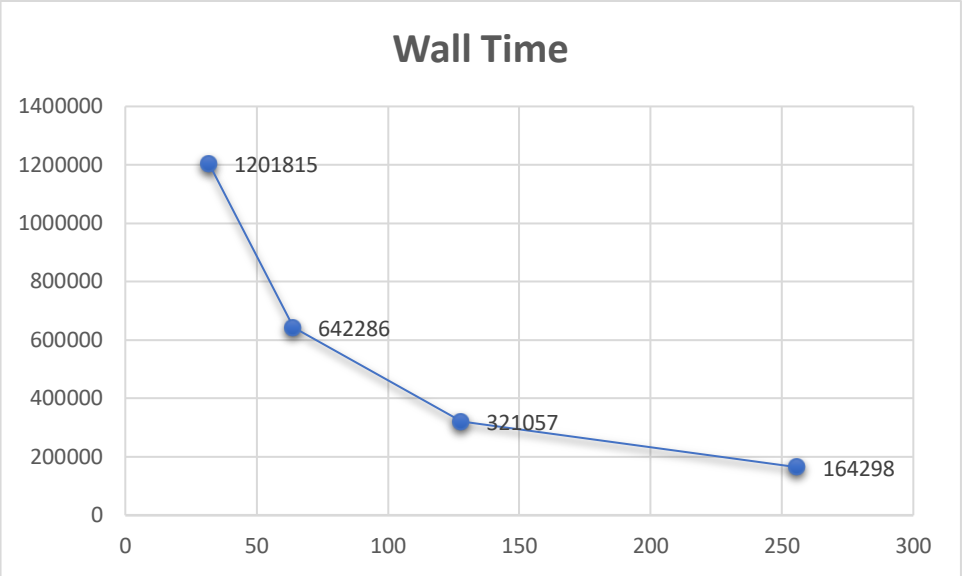
64:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_192.jdk/Contents/Home/bin/java ...
Total number of requests sent: 359520
Total number of successful responses: 359520
Total number of failed responses: 0
Wall time is: 642286 milliseconds
----------------------------------------
The mean of all latencies: 86 milliseconds.
The median of all latencies: 78 milliseconds.
The throughput: 0 milliseconds.
The 99th percentile of latencies: 211 milliseconds.
The max response time: 1708 milliseconds.

Process finished with exit code 0
```

32:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_192.jdk/Contents/Home/bin/java ...
Total number of requests sent: 360016
Total number of successful responses: 360016
Total number of failed responses: 0
Wall time is: 1201815 milliseconds
----------------------------------------
The mean of all latencies: 79 milliseconds.
The median of all latencies: 75 milliseconds.
The throughput: 0 milliseconds.
The 99th percentile of latencies: 161 milliseconds.
The max response time: 1548 milliseconds.

Process finished with exit code 0
```

Then we have the graph for the wall time by the number of threads:



The graph of throughput and mean response time against number of threads: