the URL for your git repo.

<https://github.com/jerryyummy/distributed_ssyystem_assignment1.git>

a 1-2 page description of your client design. Include major classes, packages, relationships, whatever you need to convey concisely how your client works.

**Firstly, the Client class initialize with an ip address, which is the url for the website we want to request. And it has get() method and post() method to create a request. I use the CloseableHttpClient class to create HttpClients.createDefault(); the use HttpGet get an instance of Get() request.and the client will excute this instance , which will returns the response. From the response, we can get the status code, response body. By the way, the Post() uses the same process.**

**When I want to create multi-thread program, I create a class called ThreadGroupTest to refer the client and receive parameters. I create a threadpool by ExecutorService executor = Executors.newFixedThreadPool(100000), and this submit() method will generate a thread, for every thread in the pool, I use CountDownLatch latch = new CountDownLatch(threadGroupSize) so it can create enough thread, and it will get() and post() per 1000 times. The timestamp will record how many seconds it takes, and the total time, throughtout. Use Thread.sleep(delaySeconds \* 1000) to set a timeout when a thread finishes 2000 times requests.**

Client (Part 1) - A Plot show the throughput for the tests comparing the two servers. This should also include a screen shot of your output window with your wall time and throughput for each of the 6 tests.

A screen shot of a computer screen

Description automatically generatedA number on a black background

Description automatically generatedA screenshot of a computer

Description automatically generated

A number on a black background

Description automatically generatedA black background with white numbers

Description automatically generatedA black screen with white text

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | java | go |
| 10,10,2 | 53 | 46 |
| 10,20,2 | 71 | 71 |
| 10,30,2 | 107 | 49 |

a plot comparing the throughout

As we can see the java server’s throughout is higher than go, and it costs less time. 10-30-2 for go’s result is not normal cuz I justify the thread pool size in case of overload. But I tried many times again and I found go is faster than java, maybe there is something happened in threadpool during lab

Client (Part 2) - run the client as per Part 1, showing the output window for each run with the specified performance statistics listed at the end, and a plot comparing the two servers.

Go:

A screenshot of a computer

Description automatically generated

10-10-2

mean response time (millisecs) 87

median response time 86

p99 (99th percentile) response time 84

min and max response time (millisecs) 748 82

A screen shot of a computer screen

Description automatically generated

10-20-2

mean response time (millisecs) 91

median response time 88

p99 (99th percentile) response time 82

min and max response time (millisecs) 740 81

A screenshot of a computer screen

Description automatically generated

10-30-2

mean response time (millisecs) 87

median response time 86

p99 (99th percentile) response time 84

min and max response time (millisecs) 777 82

java:

A screenshot of a computer

Description automatically generated

10-10-2

mean response time (millisecs) 183

median response time 181

p99 (99th percentile) response time 176

min and max response time (millisecs) 1334 172

A screenshot of a computer

Description automatically generated

10-20-2

mean response time (millisecs) 182

median response time 178

p99 (99th percentile) response time 174

min and max response time (millisecs) 914 171

A screenshot of a computer screen

Description automatically generated

10-30-2

mean response time (millisecs) 184

median response time 180

p99 (99th percentile) response time 176

min and max response time (millisecs) 871 172

The plot of your throughput over time for a single test

A blue line graph with a point

Description automatically generated

Java

A graph with a line drawn on it

Description automatically generated

go