Gail Dylan Salak
CSE 130

Load Balancer Write-Up

Testing:

- I began testing with multiple curl requests ending with "&" in a .sh script to test if my server could handle multiple connections at once.
- I then moved on the git tests to see how my server could improve. I failed a lot of them.
- I was able to find a few test scripts on Piazza and Discord:
    - shared test git repo from Piazza (zachjicha git repo)
        - These tests were separated into classes which test different aspects of the program. These classes included args, get, put, head, multi-thread, server-down, and load.
        - Passed most of the tests. Hang on server-down4.test
- Disclaimer: I fail two tests on git as of right now. I'm unsure why. Changing the X value seems to fix one test but make another fail.

Questions:

For this assignment, your load balancer distributed load based on number of requests the servers had already serviced, and how many failed. A more realistic implementation would consider performance attributes from the machine running the server. Why was this not used for this assignment?

To be fair, when this assignment was assigned, we had not yet gone over performance, so that would have been very difficult. We don't know how to examine or manipulate the pipeline in ways to decrease latency. We have been increasing throughput through multi-threading, but other performance concepts are very challenging as of right now.

This load balancer does no processing of the client request. What improvements could you achieve by removing that restriction? What would be cost of those improvements?

If we could so even some simple processing of the client requests, we could then do some error checking for erroneous requests. Then, instead of sending the message to the server and waiting for the server to respond with an error, we can immediately send a 400-message back to the client without sending it to the server at all. These improvements would come at the cost of the loadbalancer doing more work per request as it would parse every single message. In addition, it creates more room for errors in the loadbalancer itself.

Using your httpserver from Assignment 2, do the following:

– Place eight different large files in a single directory. The files should be around 400 MiB long.
– Start two instances of your httpserver in that directory with four worker threads for each.
– Start your loadbalancer with the port numbers of the two running servers and maximum of eight connections.
– Start eight separate instances of the client at the same time, one GETting each of the files and measure (using time(1)) how long it takes to get the files. The best way to do this is to write a simple shell script (command file) that starts eight copies of the client program in the background, by using & at the end.

I was not able to figure out how to use time. First, I did not have enough space in my VM to create eight 400 MiB files. Then, when trying to test with time(), it was not working. I tried to restart the VM, but now it is not even starting back up. I must give up in order to submit on time.

Repeat the same experiment, but substitute one of the instances of httpserver for nc, which will not respond to requests but will accept connections. Is there any difference in performance? What do you observe?

Same as above.