WRITEUP.pdf

As with all programs when I add new code I check if it functions properly by testing it.

My first goal was to get the port forwarding functionality working.

I edited the original starter code so that it would be able to pass on the request to another server and be able to read back a response. Once I got this aspect working I then moved on to the functionality of it picking the correct port.

I would test this to see if it alternated ports for each request.

I additionally added threading to my program so that it could handle multiple requests at once. For this, I used an underlying queue structure. I created the struct and functions for a global queue and did unit testing to ensure that those functions worked. I then tested adding requests to the queue and then removing them to ensure my multithreading was efficient.

Once I had completed the multi threaded aspect of my program, I then moved on to the periodic health checks. I tested this by breaking and restarting servers to check for all cases.

• 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?

Because we are running all the servers and load balancer on the same machine we do not need to test for traffic, latency, and hardware. We would not expect the performance to be vastly different. Additionally we don’t not have the resources to more holistically test the server.

• 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?

Because the load balancer and the servers are running in the same place, we can preprocess the data in the load balancer or check for errors and only send the valid request to the httpserver. You can improve fault tolerance by making these improvements however we would increase the complexity of the assignment. We could also use the load balancer for batching purposes.

• 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.

• 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?

I observe that the output returns slower as when we run nc on one of the servers we have to wait for the server to time out. Additionally we have one less server to forward requests to. The performance is slower.