Introduction

I will be referring to the self implemented load balancer as stage 1 and the one using swarm as stage 2.

Both stage 1 and 2 follow the same pattern: When the amount of HTTP-servers increases, the mean response times get lower. And when the amount of threads sending HTTP-requests simultaneously increases, the mean response times grow. They both work just as you would excpect, and that is how they are supposed to work. In stage 2 these results can clearly be seen: No matter how many threads are running, the implementation with 5 HTTP-servers has always lower mean response times than the one with 3 servers and the one with 3 servers has lower response times than the one with just 1 HTTP-server. With 20 threads running in parallel, the 1 server solution reaches a mean response time of 95.7 ms, 41 ms for 3 servers and 34.4 ms for 5 servers.

For both stages 1 and 2, the solution with just 1 server running has clearly slower response times than the other solutions, but the solutions with 3 and 5 servers running do not differ that much from each other. As said earlier, the response times are slower with just 3 servers in stage 2, but in stage 1, they are basically the same. In stage 1, the 5 server solution actually had slower response times than the 3 server solution, in multiple occasions (threads: 4, 7, 9, 10, 11, 13 and 15). In stage 1 the different solutions reached the following mean response times with 20 threads running: 82.8 ms with 1 server, 47.6 ms with 3 servers and 44.3 ms with 5 servers.

1 server connected

Comparing stages 1 and 2 show that in stage 1 the mean response times were in fact shorter with just one server running, than the similar solution in stage 2. Not only with 20 threads, but on every iteration after 4 threads. This must be the result of my load balancer approach not doing anything other than receiving the GET-request and redirecting it forward, aswell as passing the response through the loadbalancer to the client. While in Swarm there is probably some additional network traffic even with just one server connected. This traffic is most likely some sort of ”server ready to receive data” messages back to the load balancer server.

3 servers connected

The difference between the 3 server solutions were not that big either. In both of the stages, the mean response times were lower than 40 ms until the 17th thread. After the 17th thread though, the response times started to get lower in stage 2, while in stage 1 they kept growing. The same thing happened in stage 2 between 8 threads and 15 threads, while in stage 1 the response times grew when ever the amount of threads grew.

5 servers connected

The 5 server solution in stage 2 was clearly better than the one in stage 1. It was looking good until the thread number reached 9 with both stages having response times just over 20 ms . After 9 threads, the response times in stage 2 started to grow clearly slower than the ones in stage 1. At 15 threads running, the response times in stage 1 were already over 40 ms, while in stage 2 they were only around 25 ms.

Overall

Overall comparing my load balancer solution to the swarm’s load balancing, it seemed like the response times just kept growing in my solution as the amount of threads grew. In Swarm’s case they also grew, but not nearly as fast (except with just 1 server). Looking at the graphs of both stages, the graph of stage 1 is very straight forward: It keeps going up when the amount of threads grow on each of the solutions. In stage

2 however, especially in the 3 server solution, the graph sometimes even goes down, and then slowly back up.

Other remarks

I noticed that benchmarking the Swarm solution was way slower on each occasions, than benchmarking my own load balancer implementation. I quickly discovered, by following the CPU usage of each of the servers, that in Swarm implementation, the CPU usage would go to 0 % on every server, every now and then. On my load balancer implementation this would not happen, but the servers would keep running at 5 % CPU all the time. I also noticed, that when I did not limit the CPU usage of the HTTP servers, the servers would still very rarely run in more that 6 % CPU in my load balancing implementation. In Swarm however, they would reach CPU usage of up to 12 %. I think this might also be because of some additional network traffic being sent and handled by the load balancer in Swarm.

Even though this project was extremely time consuming for me, I have to say I learned so much new things doing it. Thank you for that.