CS6847 : Cloud Computing - 1

Pruthvi Raj R G

February 2020

1 Overview

In this assignment, we get acquainted with the autoscaling feature available on the AWS platform. Autoscaling helps clients automatically scale up or scale down their applications running on the cloud-based on the load on the server. The autoscaling feature helps reduce excessive costs incurred in hosting the server and gives the flexibility to run the server.AWS Auto Scaling monitors your applications and automatically adjusts the capacity to maintain steady, predictable performance at the lowest possible cost.

In this report I record my observations and test out the autoscaling feature in aws. I also anlalyze the variation of different parameters like Network I/O , Memory and CPU Utilization during the testing.

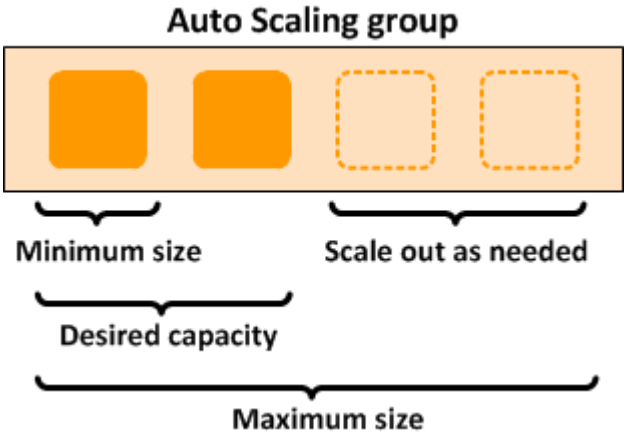

Figure 1: Pictorial representation of auto-scaling

2 Idea for load testing the instances

I looked out for online tools for testing out the serer using variable request rates. However, I could not find any testing tool that provided an interactive testing tool in which we could explicitly mention the rates of requests. Hence I decided to go with Apache Bench, which allowed me to indicate the number of requests to be made and the **concurrency** of the requests to send, i.e., the number of requests that would be sent at a single point of time. I kept the number of requests equal to the concurrency so that all requests are sent at once to the server. I then created a bash file to change this value of concurrency using a simple for loop.

I plotted a graph between the response time of each request to know the variation of response time with a change in concurrency(request rate). I observed that at all the different rates, there was a **warm up period**, as illustrated in the graph. If I took the average of response time directly for each request rate, this warm-up period would skew the metrics as the warm-up period would increase with an increase in request rate. Therefore, To normalize for this warm-up period, I took an average of only the **last 50-90 percent** of the responses(as they represent the request rates much better than the initial points.)

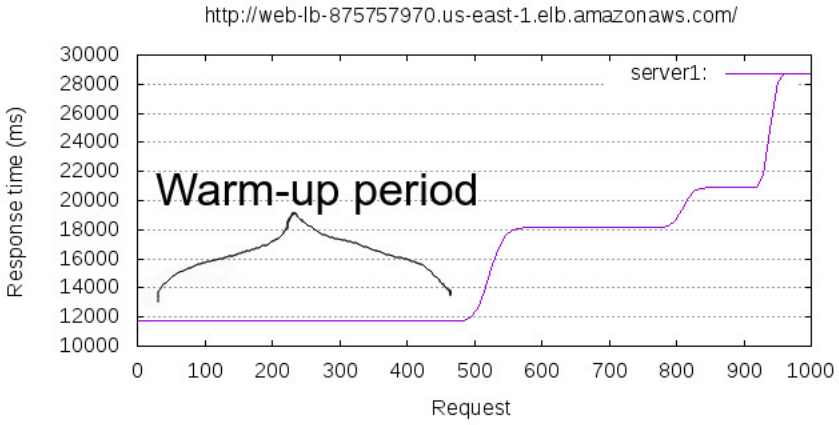

Figure 2: Indication of the warm-up period in the response time

3 Experinment on single instance (Without Autoscaling)

The details of creating an instance on the aws could has been explained in detail in the accompanying ReadMe.md file. Below I show some of the response charts that I got for different request rates.The maximum concurrency that I could send was 8000. I then plotted histogram on an online website.

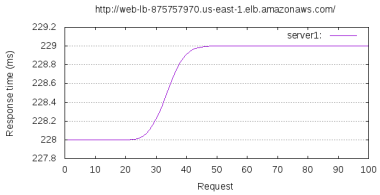

Figure 3: Variation of response time with request rate as 100

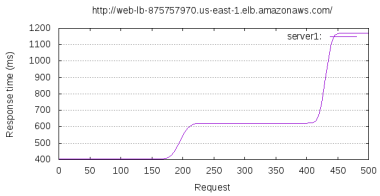

Figure 4: Variation of response time with request rate as 500

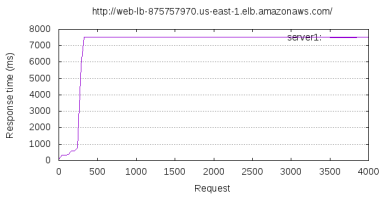

Figure 5: Variation of response time with request rate as 4000

I show the server program I wrote below. It is a simple program witch contains for loop inside another for loop and calculates the sum of random numbers for random amount of time. **I chose this function as this is a close representative of the actual server where different client need different amount of time on the server based on a hidden parameter.**

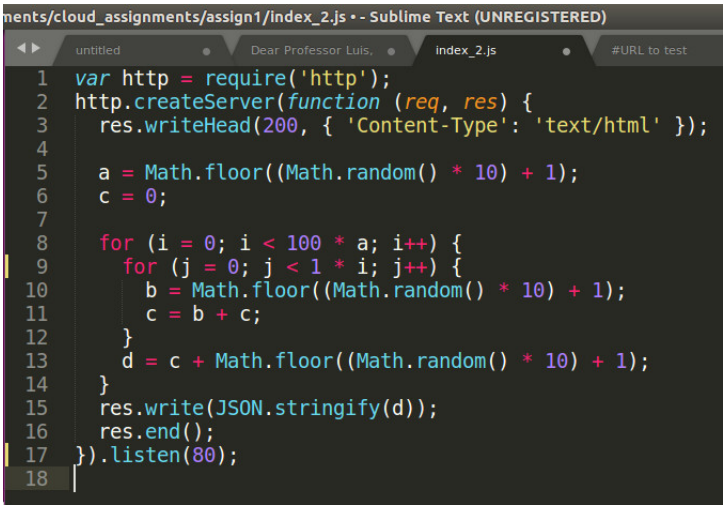

Figure 6: Server function running on the instances.

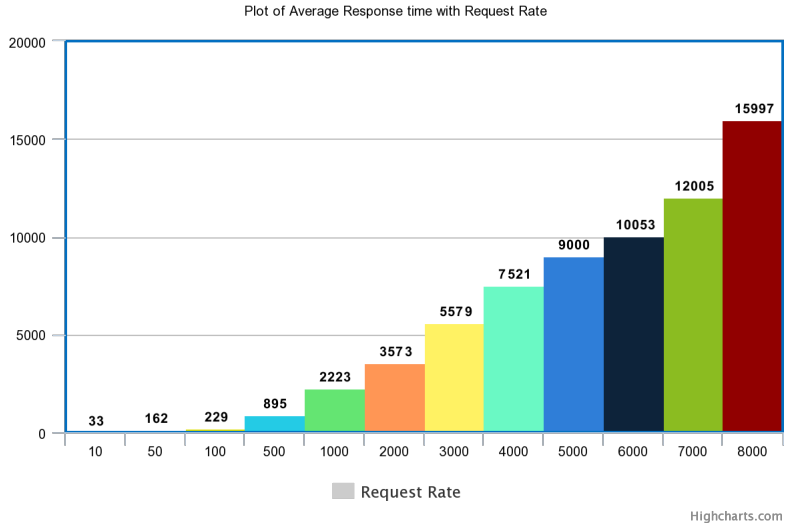

Figure 7: Plot of Average Response time with change in request rate

4 Experinment on Autoscaling

The details of creating an autoscaling on the aws could has been explained in detail in the accompanying ReadMe.md file. Below I show some of the response charts that I got for different request rates. I created 3 instances and I kept the CPU utilization threshold as 24 percent.

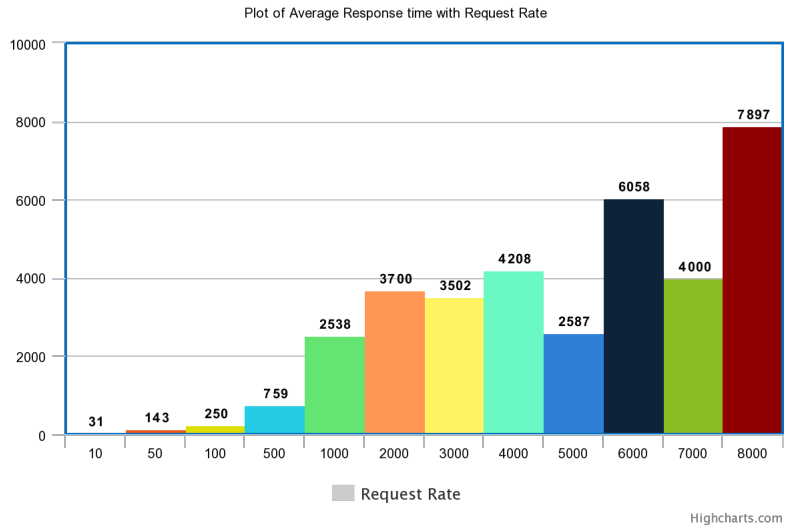

Figure 8: Plot of Average Response time with change in request rate

5 Comment about various parameter during autoscaling

I observed that the cpu utilization increased with increase in request rate. I had kept the cpu utilization threshold as 24 percent. Hence as the utilization increased it lead to creation of new instance around the request rate = 4000. Also I expect that one more instace got created around the request rate = 7000. The Network parameter closely follwed the CPU utilization however the memory usage almost remained constant.

I also include some of the screenshots during the creation of instances.

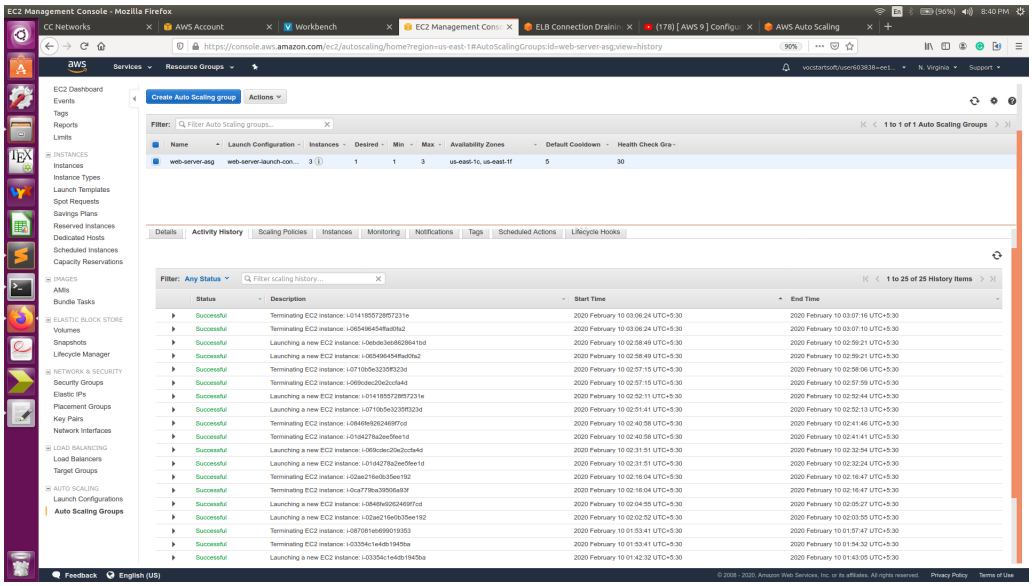

Figure 9: -

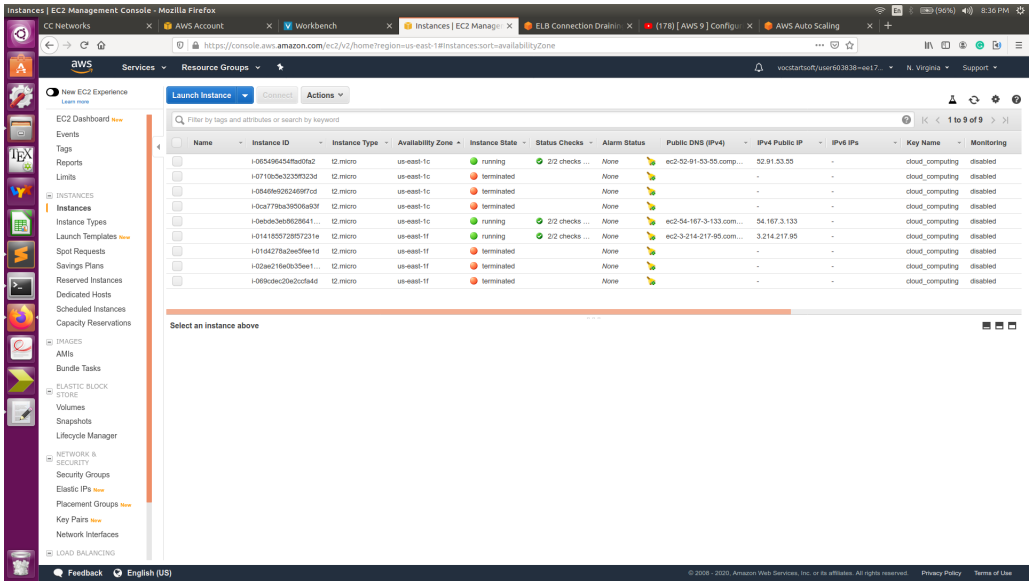

Figure 10: -