DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, IIT MADRAS

CS6847: Assignment 1

Harshit Kedia
CS17B103

1 LOCAL SERVER AND CLIENT

- At first, Server and Client programs are run on the same machine(Local host-client).

Request Rate	Response Time
10	0.004862594604492188
100	0.0002662084102630615
1000	0.0002327509880065918
10000	4.3746998310089114e-05
20000	4.6874409914016724e-05
100000	3.273630571365356e-05

- At local host, we can see that even for 100000 packets, we have very low response time. Surprisingly, the response time initially decreases with increasing rate.

2 SINGLE INSTANCE SERVER

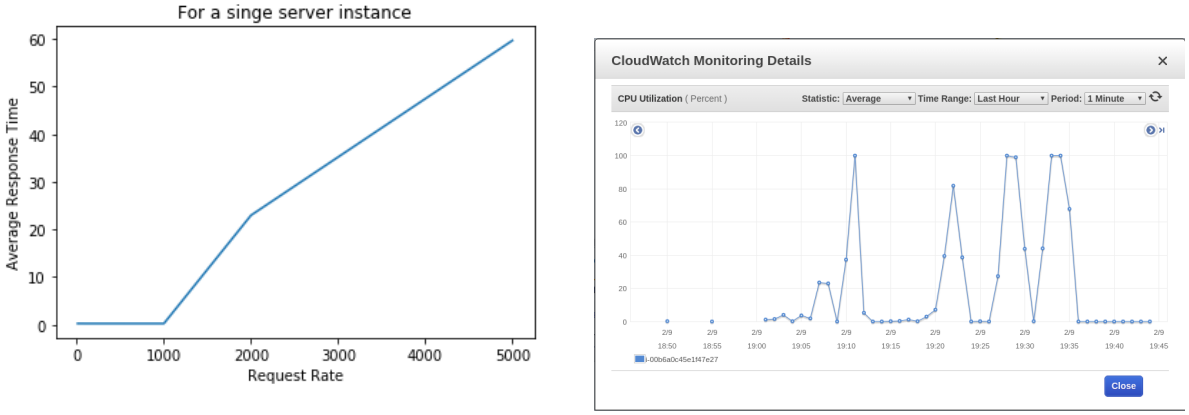

Figure 2.1: Response time and CPU usage plots for single instance server, upto request rate of 5000 per second for duration of a few minutes.

- The server side task chosen for this assignment is computational as well as memory extensive. Hence, we can see that even with about 500 requests made, the server bottle-necks and reaches 100% CPU utilization. The peaks we observe in the graph are of rates 10, 100, 1000, 500, 2000

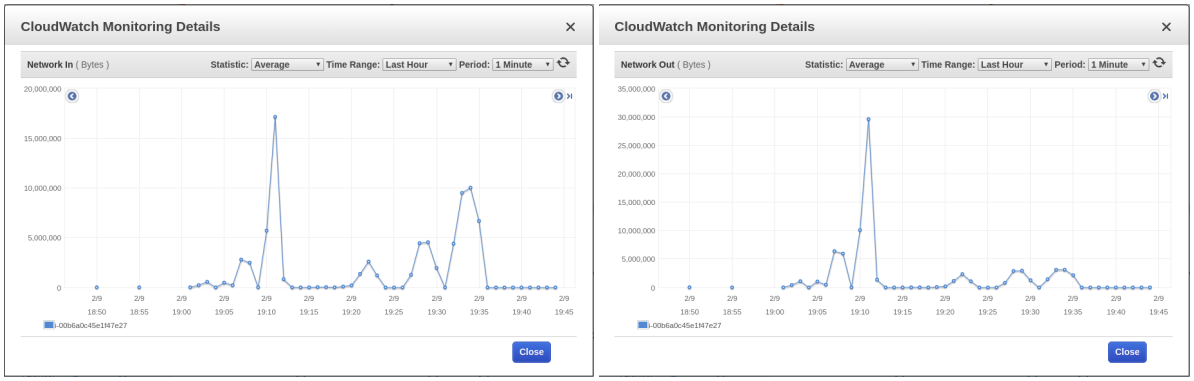

Figure 2.2: These are the plots of network in and out activity for a single instance server on a 1-hr long x axis scale, hence we only need to focus on the second half of the curves,where my testing is done.

and 5000 respectively starting from the very tiny left-most peak. Note that all of the clients run for a fixed amount of time. The flat-end of the peak shows the queue buildup getting processed and hence wider peaks at higher rates.

- From the network activity graph we can conclude that the maximum rate at which the server can compute and reply back is mutually limited to around 3 million bytes/second. Hence, if we force upon more requests, the queue of tasks to do gets built up and hence, the overall response time for the server goes high.
- The response time of individual packets is monotonously increasing and saturates just for the last few packets in this setup.

3 Cloud Server with Auto-Scaling

Rate of Requests	Avg CPU Usage	No of Instances	Response Time
10	1.5%	1	0.2870
100	10%	1	0.2836
1000	50%	2	0.2779
5000	70%	4	14.78125

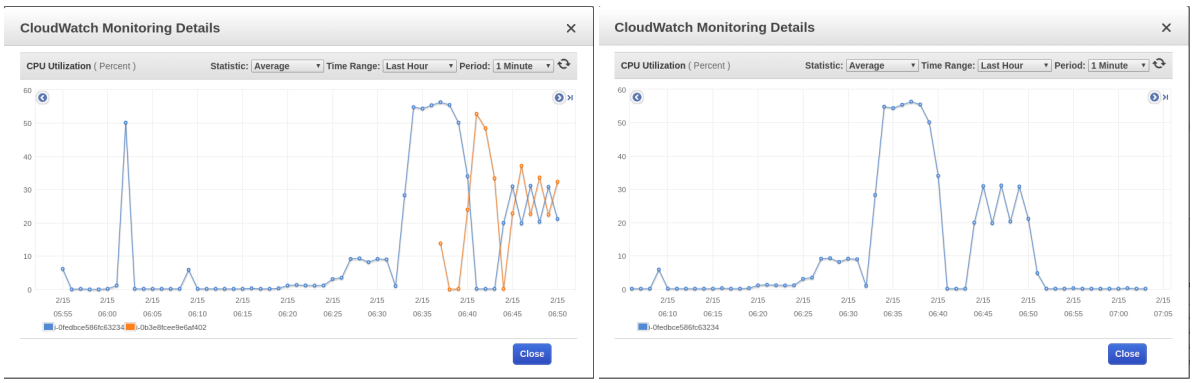

Figure 3.1: For request rate of 1000, Load Distribution before and after auto-scaling. Second image clearly shows that the workload is almost halved as soon as 2nd instance comes up.

- Key points to remember during setup: Creating a valid image, Shell script in launch configuration, scaling policies, listening and target ports of load balancer and ultimately a http server, "/index.html", to respond to health checks.

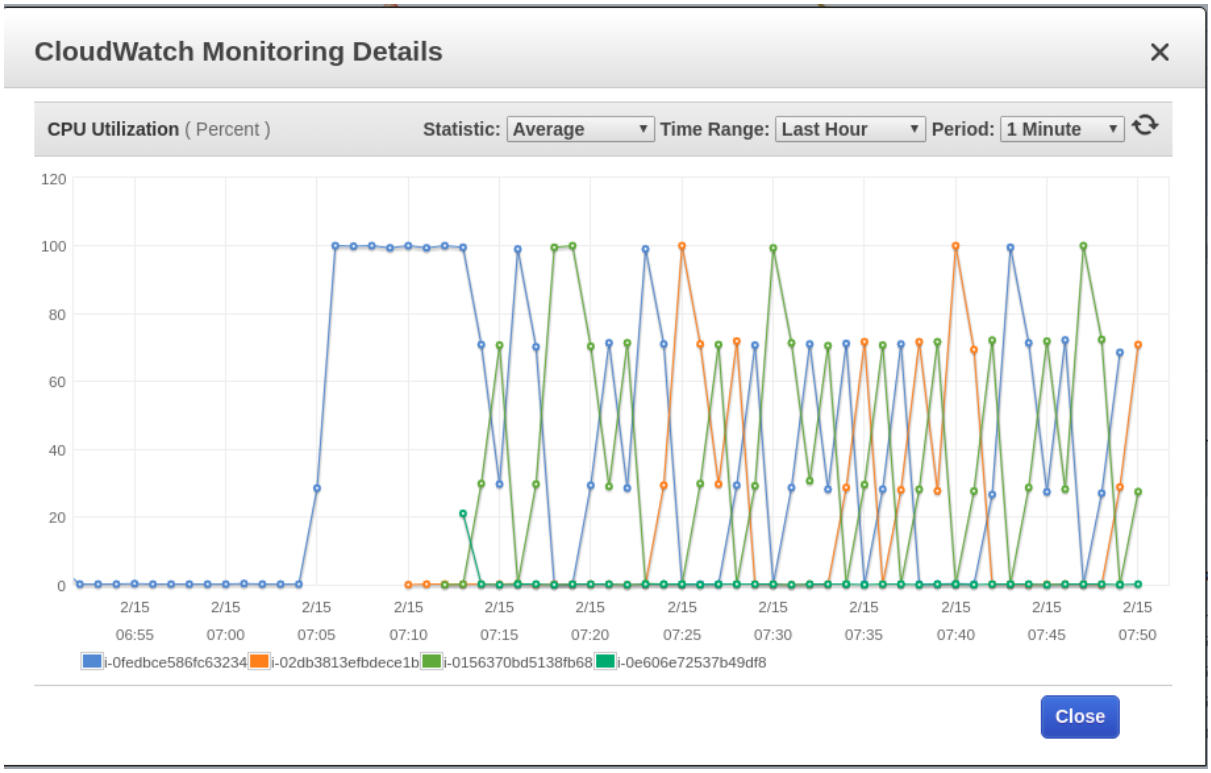

Figure 3.2: CPU usage across 4 instances for request rate of 5000 per second.

- The policy used for auto-scaling here is Avg. CPU usage being more than 50% for a one minute period. The time taken by new instance to come up and pass the system as well as load-balancer health check is quite significantly high (7-8 mins).
- Since the get-ready period of the instance is so high, the client program sends request constantly for about 20 minutes.
- On closely looking at the CPU usage plot (Fig:3.2), we can see that there is some kind of uneven distribution of load across the instances. The 4th instance hardly has any workload (<1%), whereas, the other 3 have some kind of scheduler type workload varying from about 60% to even 100%.
- The best efficiency achievement can be observed in the case of around 1000 requests per second. As we can see in the graph for Single instance, it is around this rate where the server starts to choke. Whereas even at 1500, the response time is very consistent and low in case for 2 instance load balance.
- Since we have marked the CPU usage to 50% for auto-scaling, we see a sudden decrease of response time whenever a new instance comes up. Also, since the new instance takes so much time to ready up in AWS, till then the CPU is used at 100% and hence the average is as high as
- In fig-3.5, we can easily see the number of instances running for the server. Also observe that booting up and termination can take as high as 10 minutes.
- Quite clearly, the network usage is evenly distributed across most of the instances, Also, it is very similar to the CPU usage graph for the respective instances. Hence, the workload and bandwidth required is proportional, and thus, scaling based on network usage or CPU usage is analogous.
- Request rates more than 5000 were getting bottle-necked by my own laptop WiFi connection, i.e the effective rate is lower only.

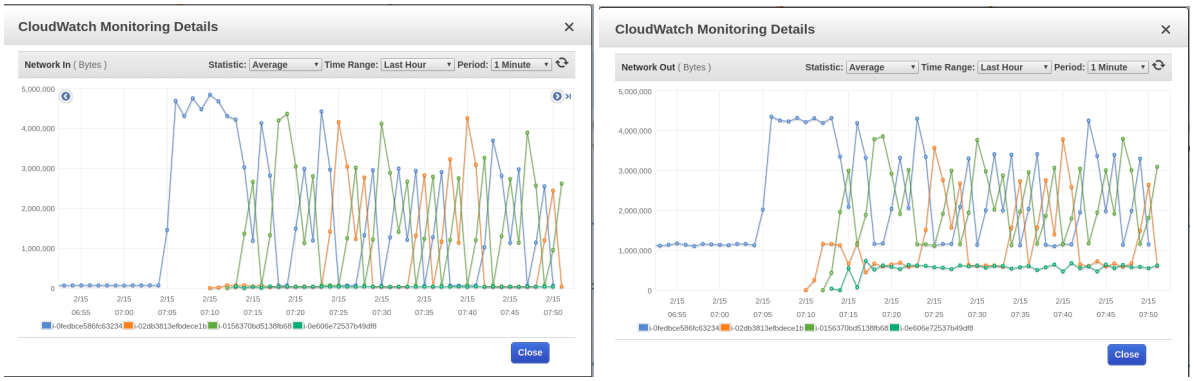

Figure 3.3: The network in and out usage for all 4 instances for request rate of 5000.

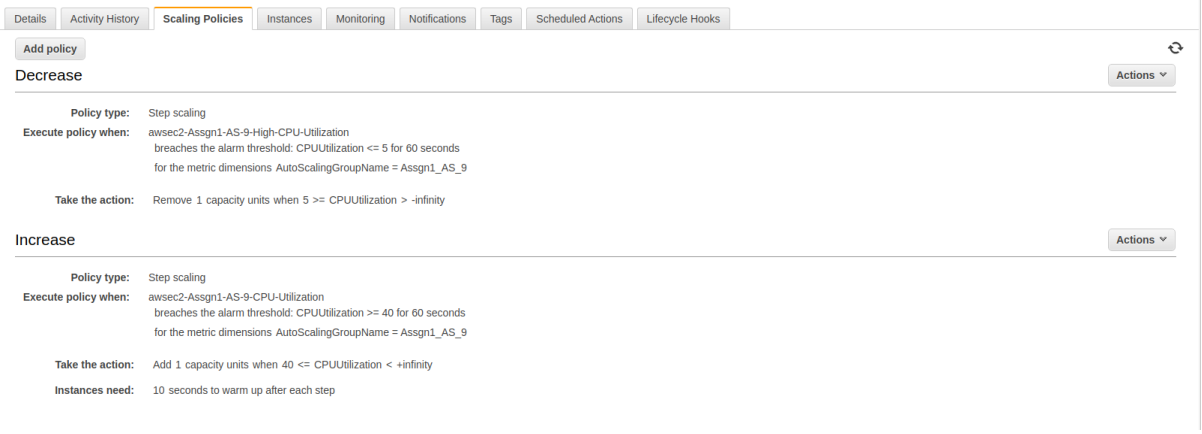

Figure 3.4: The Scaling policy used for most experiments done on auto-scaling.

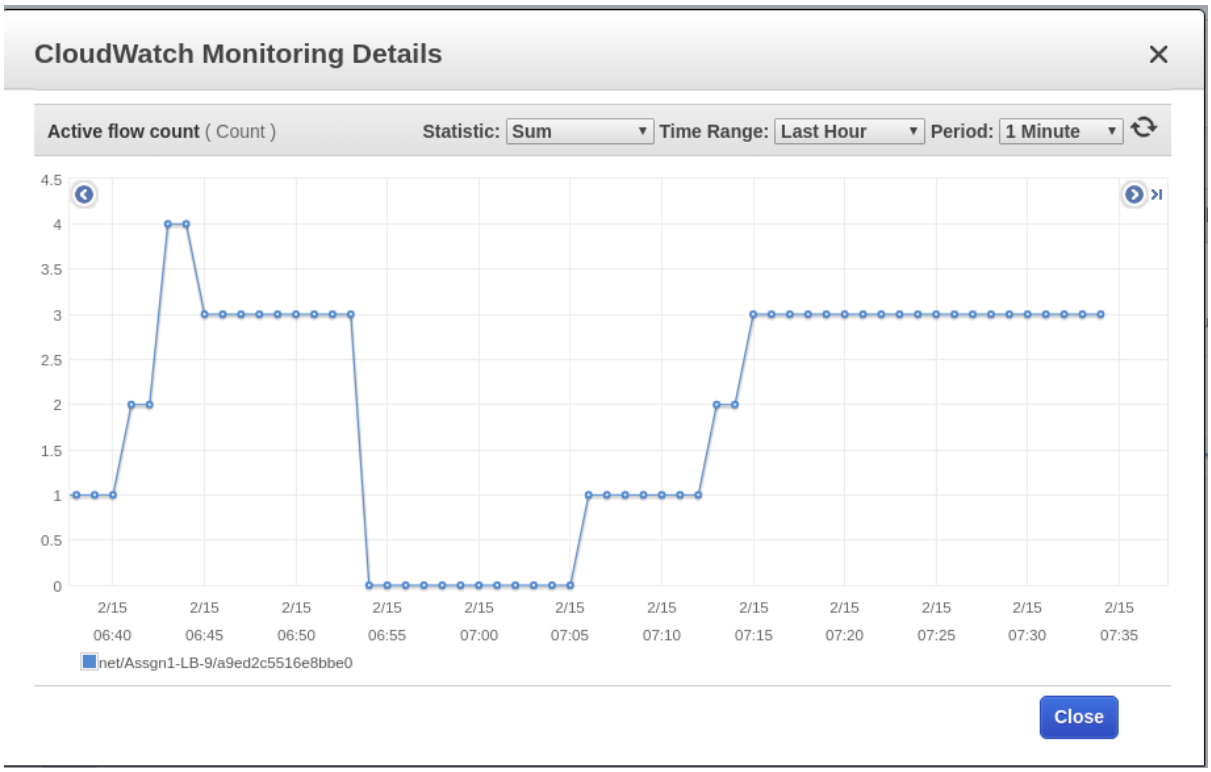

Figure 3.5: The dynamic start and termination of instances during testing.