Project 2

Price/Performance of EC2 for Serving Dynamic Web Content

a .Group Name: SWAT Kats

Group Members: Dharini Chandrasekar, Surya Kiran Valiveti

We performed our analysis for 7 instance types. They were:

t2.micro
t2.small
m2.medium
m3.medium
m3.large
c4.2xlarge
c4.4xlarge

b. The most computationally expensive request was that with a parameter of 500 (Generation of first 500 prime numbers) by 10 users each user looping 15 times in a time interval of 1 second.

The lowest latency was observed to be given by: **c4.4xlarge**

The least computationally expensive request was that with a parameter 20 (Generation of first 20 prime numbers) by 10 users looping 20 times in an interval of 1 second.

The lowest latency for this was observed to be given by: **c4.4xlarge**

c. The most expensive configuration we implemented was a **c4.4xlarge** large instance type with the cost of running being $0.928 per hour. This had a 16 vCPUs and 30 GiB of RAM.

This was indeed a beast and provided a very good performance for the computationally intensive tests with an average response time (for the 10 parameters considered) for the requests to be around 120 milliseconds.

Compared to a c4.2xlarge instance, the cost had increased by 100% (doubled) but the performance increased only by 35 %. Therefore, even for a server running computationally intensive task, the response time of 200 MS is acceptable therefore it didn't deliver the performance that would be expected from its pricing.

If we were running a simple site, we would prefer a general purpose **t2.small** instance type as it meted out a decent performance for the least expensive requests (around 63 ms).

For a complex site, we consider the average response time as a benchmark to decide the instance type. The configurations - t2.medium, c4.2xlarge and c4.4xlarge all had an average response time of less than 500ms. So if cost was a constraint then we could go with t2.medium which performs decently for the price it is being offered at. And for obtaining a lower response time on computationally expensive requests, we would go with **c4.2xlarge.**

For the calculation of cost/request we have assumed that the simple server is being hit by 100 requests per second. The cost of our simple server (running on t2 small) is 2.6 cents/hour so at 100 requests per second the cost would be **0.000072 cents per request**.

d. We used **Amazon Linux AMI** for the load testing on all the instances.

e. We used **Apache (httpd)** web server for the deployment of our web page. This came pre-installed with the AMI that we chose. We updated it to the latest version for efficient performance.

f. We ran our instances in **US East (N. Virginia)** as this being geographically closely located to our test location, the response time would be better. Our Apache JMeter was installed on a system at our location **(UC campus)**.

g. Our Dynamic web page was a prime number generator that accepted from the users, a value for 'n' through a simple HTML form and generated those many prime numbers. The script for this was written in PHP with a function that computed and generated 'n' prime numbers.

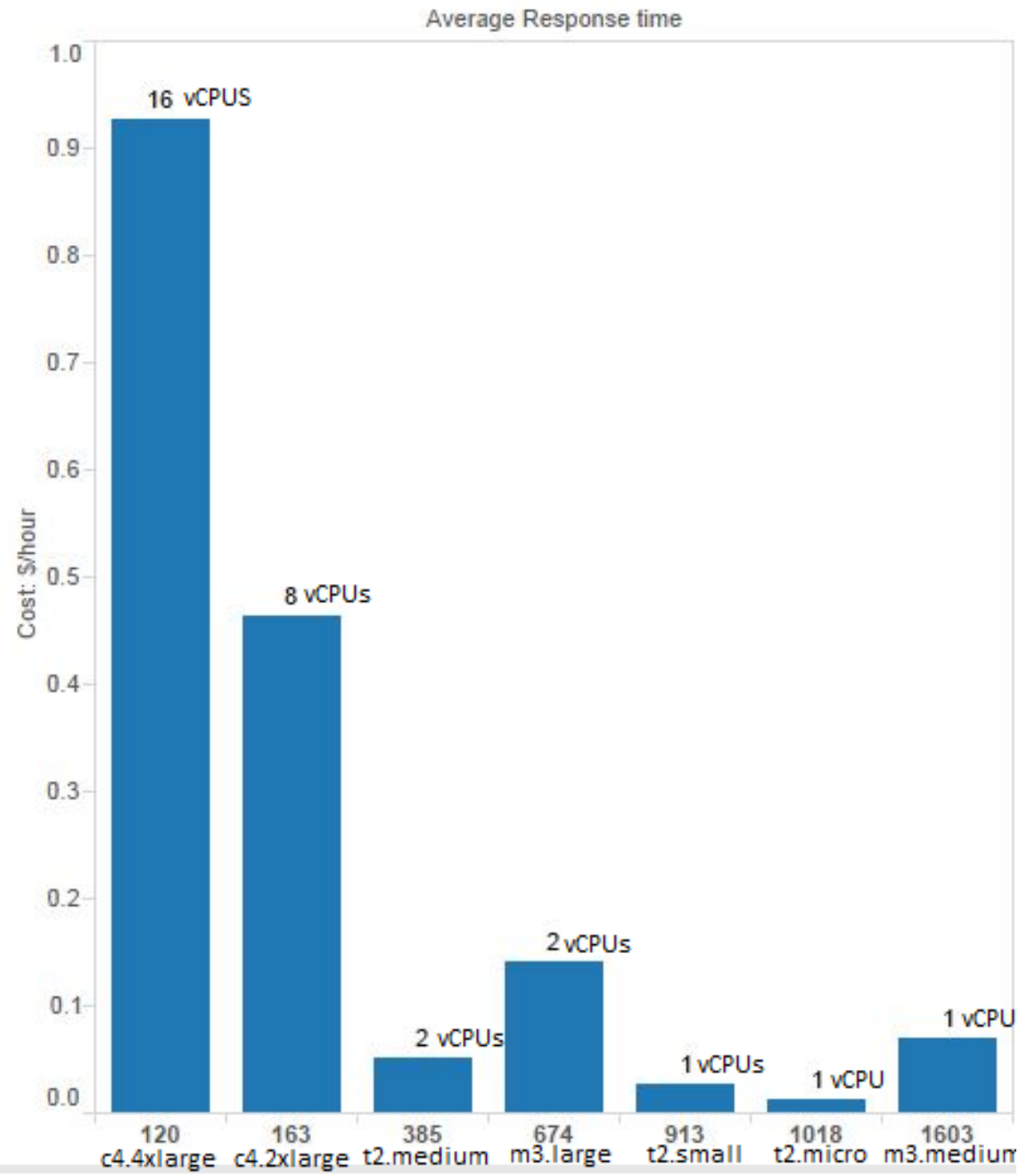
h. We set Apache JMeter to hit the website for each instance type with 10 different parameters. Each parameter had 150 requests (10 users looping 15 times). The data collected was listed in such a way that the response time (latency) was computed and displayed for each parameter value. We hit all the servers with JMeter from the same location and with the same set of parameters.

Given "n", our webpage returns the first n prime numbers. We chose 10 different values for n ranging from 10 to 500. This way we could calculate the response time of each server for each parameter. We could also analyze which instance type is best serving simple requests, which instance type had the least response time for computationally intensive requests and which server had the least average response time, keeping the cost in mind for all the instances chosen.

Taking all the above-mentioned parameters into consideration as well as the pricing we decided on **t2.medium** to be our optimal configuration/setup which is priced at 0.052$/hour.

We made an interesting observation that m3.medium, in spite of having double the cost than that of t2.medium, the performance was down by 300%. Also m3.medium had the worst performance among all the instance types' terms of average response time.

The following is plot that depicts cost/hour (on y-axis) vs average response time for 10 parameters (on x-axis) with the instance type details. An interesting observation is that more the number of CPUs and more the price need not mean better the performance which is clearly illustrated in the plot below:



i.Amazon Web Services offers many flavors for running machines. The performance largely depends on the overall configuration and processor type than just the RAM size. So one could take this into consideration and not just go by the factor "greater the price – better the performance" as this has been proved wrong by our test analysis.