International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Experimental Analysis of Performance Metrics for Configuration of Auto Scaling Groups

Dhanush ND[1], Abhishek Deshmukh[2], Gowthami M[3]: Arunkumar P Chavan [4], Dr. H. V. Ravish Aradhya[5], Nagendra N N[6]

[1]Dhanush ND, Department of Electronics and Communication Engineering, RV College of Engineering, Karnataka, India.
[2]Abhishek Deshmukh, Department of Electronics and Communication Engineering, RV College of Engineering, Karnataka, India.
[3]Gowthami M, Department of Telecommunication Engineering, RV College of Engineering, Karnataka, India.
[4]Arunkumar P Chavan, Assistant Professor, Department of Electronics and communication Engineering, RV College of Engineering, Karnataka, India
[5]Dr. H. V. Ravish Aradhya, Professor & Associate PG Dean, Department of Electronics and communication Engineering, RV College of Engineering, Karnataka, India.
[6]Nagendra N N, Assistant Professor, Department of Telecommunication Engineering, RV College Of Engineering, Karnataka, India.

Abstract - *Amazon web service (AWS) is a cloud computing tool used in creating different servers in different availability zones (AZ). Problem faced in allocating resource in a virtual environment is a big stumbling block for an infrastructure. The main aim of the work is to compare the CPU utilization, HTTP 5xx error rate, and queue length variation before and after implementing Auto Scaling Group (ASG). To overcome this problem, the paper employs AWS services such as Elastic Computer Cloud (EC2) where two servers were created in two different AZ to protect the user data. The complete CPU utilization of an instance might cause the service to be unavailable to the users and give out errors. This is the reason an instance family 'R5. large' is chosen which is suitable for high performance memory intensive database and which is memory optimized instance. The infrastructures fault tolerance can be measured with the help of Cloud Watch where real time data can be analyzed. There is a need for choosing appropriate value to configure ASG. To design appropriate values for configuration, a performance test was conducted, the results of the test was analyzed and arrived on the value of the threshold and maximum instances that can be auto scaled. Auto scaling group was used in increasing the instances and the threshold were previously calculated using the live stream data on cloud watch. The performance metrics like queue length, HTTP 5xx error count and CPU utilizations are compared. The performance metrics were observed to be optimum after the integration of ASG in the infrastructure. From the analysis, CPU utilization threshold set for ASG was 65% for the instance to start and 80% for instance to terminate. The performance test was conducted for approximately 500 users for stipulated duration and due results are obtained and it is used for due purpose.*

***Key Words*: Load balancing, fault tolerance, Autoscaling, Amazon Web Services (AWS), Cloud watch, Cloud computing, Elastic Computer Cloud (EC2) Instances.**

1. INTRODUCTION

Building a fault tolerant, reliable, highly available infrastructure is of prime importance in today's IT world. Storing and managing data on premises is an expensive task and is prone to lose of data due to any disaster. To overcome the cost limitation, and management of the data center on premises, the organizations depend on private cloud providers for services.

To develop a fault tolerant system, some of the AWS services like ASG, Cloud watch, load balancer, EC2, VPC. Cloud watch service has been used to monitor the performance parameter and based on which the threshold for ASG was set, to scale instances up or down. Depending on the load a website can handle, the maximum number of instances can be set in ASG. There is a queue for holding the requests sent by the client to the server. The increase of surge queue length implies that the application is unable to serve the incoming requests. There are spillovers if the surge queue length is full. If the queue length increases above the limit, HTTP 5XX errors are encountered, implying no more requests can be served by existing number of servers. To overcome this problem, The CPU utilization was used for autoscaling which would help in providing a highly available and fault tolerant infrastructure.

The applications of cloud are mainly in large scale and it has many different cloud services. Developing highly available application is an important part of research. To overcome this problem, FT Cloud [1] is used for developing fault-tolerant systems. FT Cloud consists of two ranks. The first one involves invocation based on components and frequencies for the purpose of component ranking. The second one is fusing the information of system structure and application designers. Once the rankings were compared an algorithm was created to search the best FT strategy.

High performance Computing is easily implemented on Cloud computing as they can make use of virtualization and Virtual machines for computationally intensive application. Here the work carried out in the paper stressed on having a fault tolerant framework for HPC for cloud. The framework proposed in the work carried out makes use of PLR i.e. Process level redundancy [2] techniques to decrease the execution time applications which are computationally intensive instead of the old technique which increases overall execution cost.

The main challenges in datacenters is, load balancing. It is observed that in real world the resource demand and workload of different Virtual machines are highly dynamic which cause inefficient migration, to overcome this problem, a load balancing scheme[3] is used to provide guarantee which is probabilistic against the overloading of machine migration, by doing this the migration overhead is reduced.

A high variety of cloud computing platform are present in the market, but in the work carried out AWS was chosen, as it is feasible and a very flexible platform [7] some of the finding of this work was to compare the different load balancers and autoscaling groups.

Mitigation of cost is done by using spot instances, the instances can be terminated without notice and hence if a lot of such spot instances are used, there is a high risk of the application going down [5]. To overcome this problem, performance analysis is made, and r5.large instance which has high CPU performance is used. There are cloud platform services like, Infrastructure as a service (IaaS) and Software as a Service (SaaS). [6] Prioritizes on IaaS and the work were carried out. But, in the work carried out there is a solution to both SaaS and IaaS infrastructure.
Various features of AWS like ELB, ASG, etc. Various features of AWS like ELB, ASG, etc. helps in a development of fault tolerant cloud infrastructure [5][9]. The autoscaling mechanisms are implemented based on the load. The reduction in user cost while choosing appropriate instance type during the autoscaling application can be done [8]. To prevent the infrastructure from crashing and overloading, autoscaling mechanism can be implemented according to the requirement of application. The autoscaling and load balancing is achieved by providing load from external software [10] and monitoring of such aspects can be carried out using various AWS services.

2. OVERVIEW OF INFRASTRUCTURE

2.1 Methodology Adopted

The AWS infrastructure is being built which is highly available and fault tolerant. Availability zones are important because if all the servers are hosted in same AZ and if that availability zone fails then all the active servers in that zone will be lost. Therefore, the servers are hosted in different AZ as to have the uniqueness and if one of the servers fails in one of the AZ then other can come up as secondary. Webserver and DB server are hosted in different availability zones and application are run on webserver. The site is hosted and readily available for the users. The CPU utilization and load is monitored.

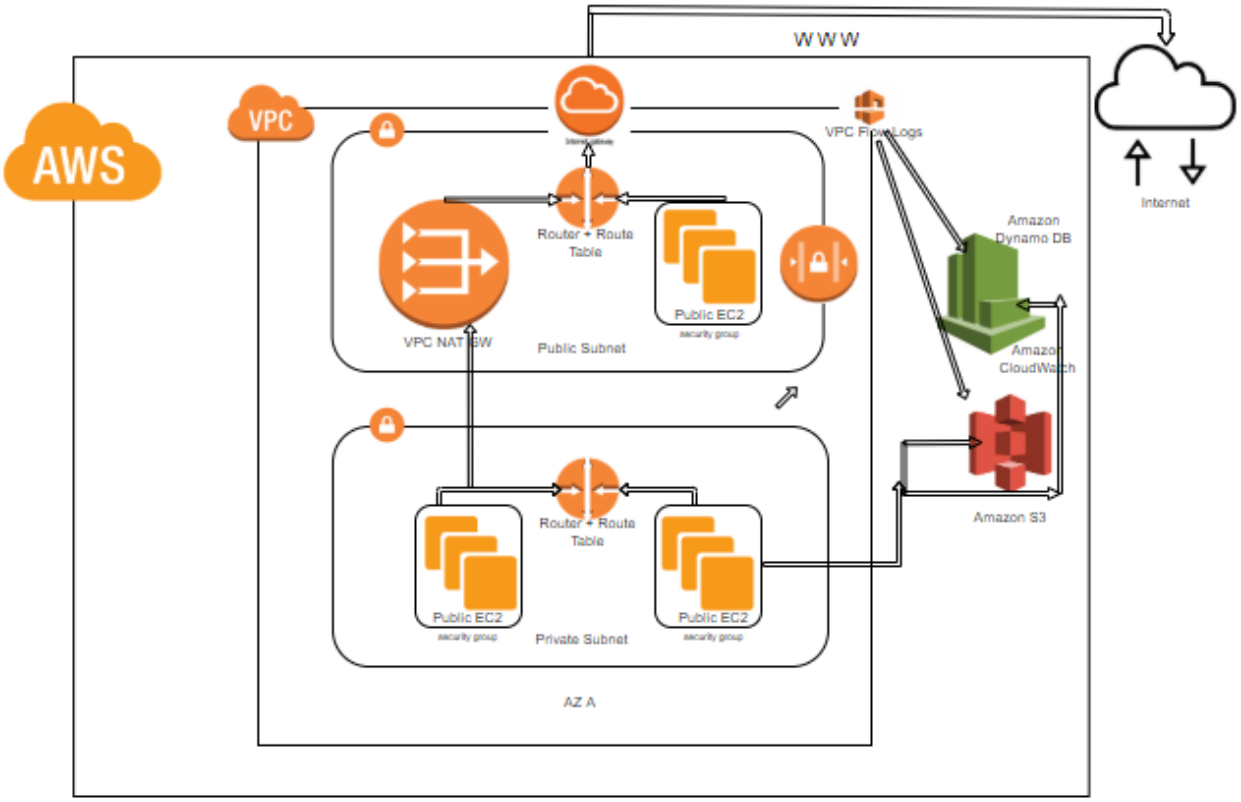

Figure 1: Block Diagram

Load balancer is attached to the servers to distribute the traffic and convert the incoming traffic from http to https. Load balancer has the specific role where threshold of the traffic is set and according to that the load is distributed. This is used for testing purposes too of the performance of the instances based on load. This load balancing plays a major role in testing the performance of servers which are to be scaled up or down based on requirement. Auto scaling is enabled to the servers such that if the threshold of the incoming traffic i.e. users is reached or threshold is reached or goes beyond the given CPU utilization set, then new instances are setup. On the contrary if the traffic goes below threshold, then the instances are terminated, hence the auto scaling. The autoscaling is set in such a way that when testing on one instance is carried out and its known that one instance is not enough for the application, then new instances are warmed up. In the process of Configuration of autoscaling groups, the threshold of running instances must be set, the family of instance to be launched is set, the number of instances which must run ,when ASG is triggered is configured, also the load balancer should be attached to the new instance. Based on conclusions of insufficiency of only one instance, the tests performed with ASG. All the monitoring of the CPU utilization and queue length is taken from logs of cloud watch.

2.2 Implementation

Step1: Creation of instances

The EC2 instance type is chosen to be r5.large [6] which has specification of two CPU's and 32GiB memory. The security groups are attached to the instance. According to the infrastructure opted same instance is created for DB server. This is the instance on which web application is hosted. The fig.2 shows the type of instance and its specifications.

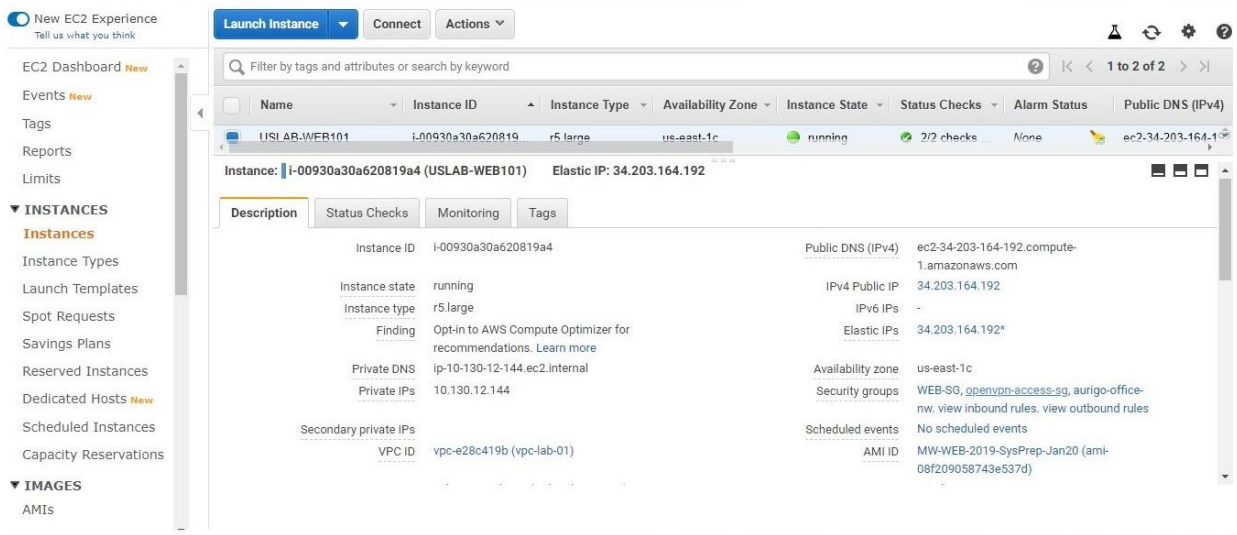

Figure 2:Instances created

Step2: Creation of load balancer

An application load balancer is created to facilitate the load balancing service provided by AWS. The created load balancer is attached to the instance created. The security groups are also attached like that of the instance. The load balancer is present within the same VPC as that of instance. The load balancer distributes the load beyond certain threshold to different instance. The monitoring service keeps the log on this load balancer. As in this infrastructure, when new instance is created after autoscaling, the load is balanced within the present running instance and newly created once. The fig 3 shows the application load balancer created for this application with a time out of 3600 seconds and various other configuration.

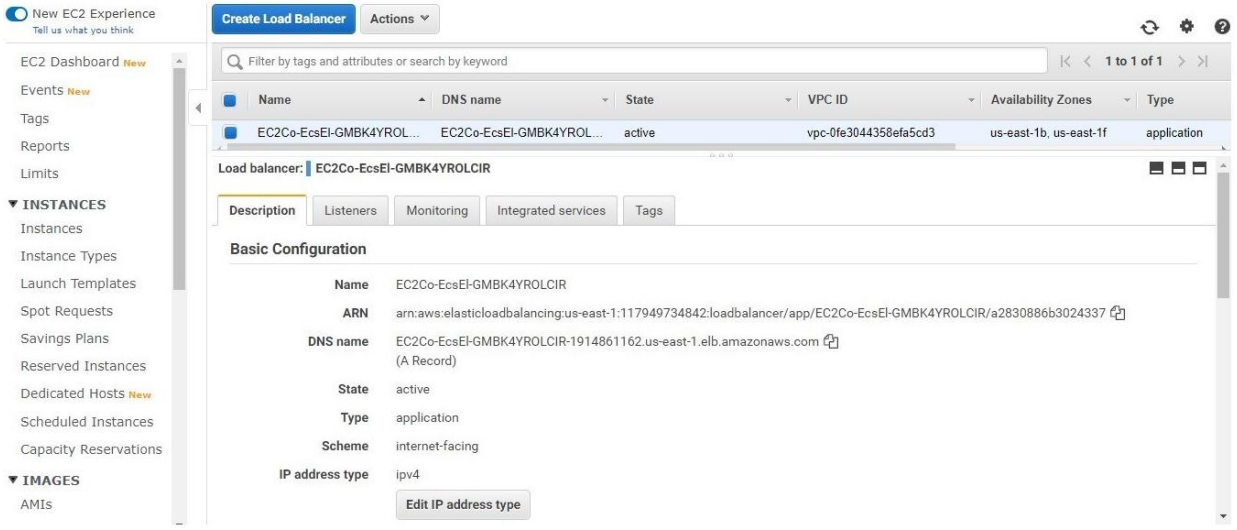

Figure 3: Creation of Load Balancer

Step3: Auto Scaling

Autoscaling group is created such that if the load on one instance reaches the threshold set, then a new instance is launched immediately. CPU utilization of running instance is the criteria to trigger ASG. The threshold of CPU utilization is set, type of instance to be launched after triggering is set and number of such instances to be created is set. Fig 4 shows the autoscaling group attached to our infrastructure.

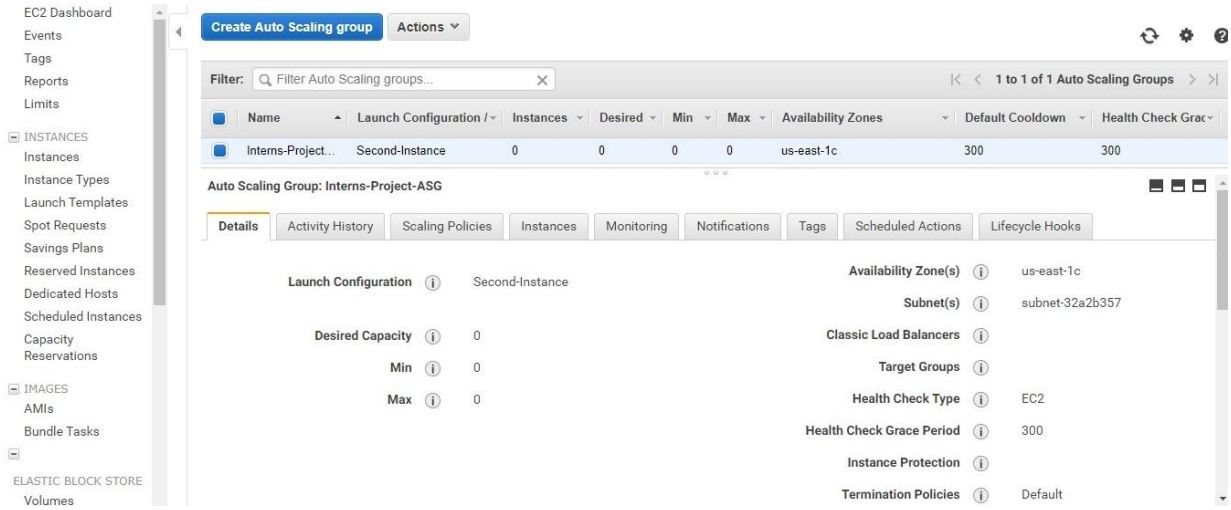

Figure 4: Creation of auto scaling group

Step4: Cloud Watch as monitoring

The CPU utilization of the running instance is monitored using AWS service called Cloud Watch. Cloud Watch is also used to get the logs from load balancer attached to the running instances in infrastructure. Using Cloud Watch graphs the surge queue is also obtained. Fig 5 shows the graphs obtained by Cloud Watch of CPU utilization of web and DB instances created in the infrastructure. The figure displays the real time CPU utilization of the instance along with the shoot up of the utilization to almost 100 percent before autoscaling.

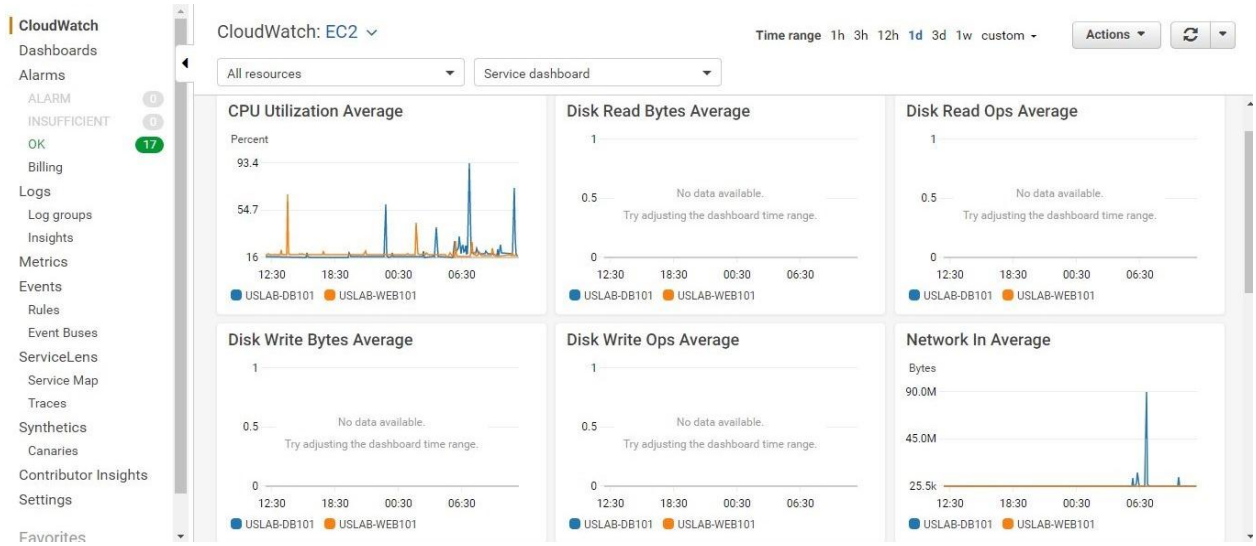

Figure 5: Cloud Watch

Step5: Generating Load

A virtual load will be generation though putties software, the terminal emulation software is hosted on a different instance from the different PC in the lab. To generate the load in putty installation of stress component in apache server is necessary. The load can be generated using the command “Sudo apt-get install stress”, “stress –c 80 –m 50 -1= 1000” hence a required load is generated as shown in the Fig.6.

```
Fetched 17.0 kB in 0s (372 kB/s)
Selecting previously unselected package stress.
(Reading database ... 51693 files and directories currently installed.)
Preparing to unpack .../stress_1.0.1-1ubuntu1_amd64.deb ...
Unpacking stress (1.0.1-1ubuntu1) ...
Processing triggers for install-info (5.2.0.dfsg.1-2) ...
Processing triggers for man-db (2.6.7.1-1) ...
Setting up stress (1.0.1-1ubuntu1) ...
ubuntu@ip-10-230-29-66:~$ stress -c 85 -i 100
stress: info: [2585] dispatching hogs: 85 cpu, 100 io, 0 vm, 0 hdd
```

Figure 6: Generation of load

3. RESULTS AND DISCUSSIONS

The performance testing was carried with a load balancer serving only one instance. Surge queue length was observed to increase as shown in the fig.7 Surge queue length shows that number of requests that are queued and waiting to be served by the instance. By the performance tests, it showed there were almost 500 requests that had queued up waiting to be served within a minute's time. This indicated that only one instance could not serve multiple requests in a short span of time, hence another instance had to be brought up to serve the limitation. Auto scaling groups is implemented to scale up the instances when there is requirement. The below Fig.8 represents the surge queue length of an instance with an autoscaling group attached to it. The difference observed was that the number of requests waiting in queue was reduced drastically when the autoscaling was implemented.

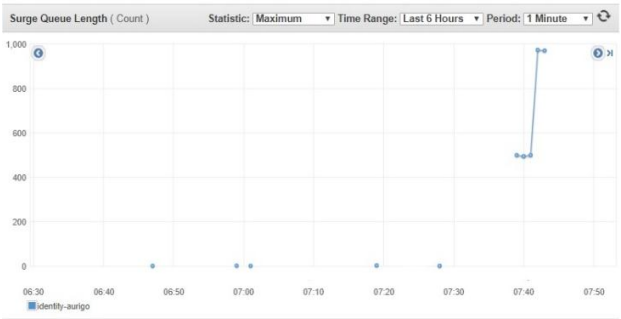

Figure 7: queue length during performance testing.

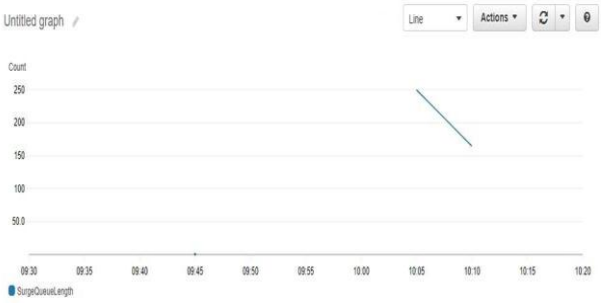

Figure 8: Surge queue length after auto-scaling

The saturation of CPU utilization of the instance means that it is no longer serviceable and starts throwing 5xx errors to users. That was obtained in the Fig.9 given by CloudWatch console.

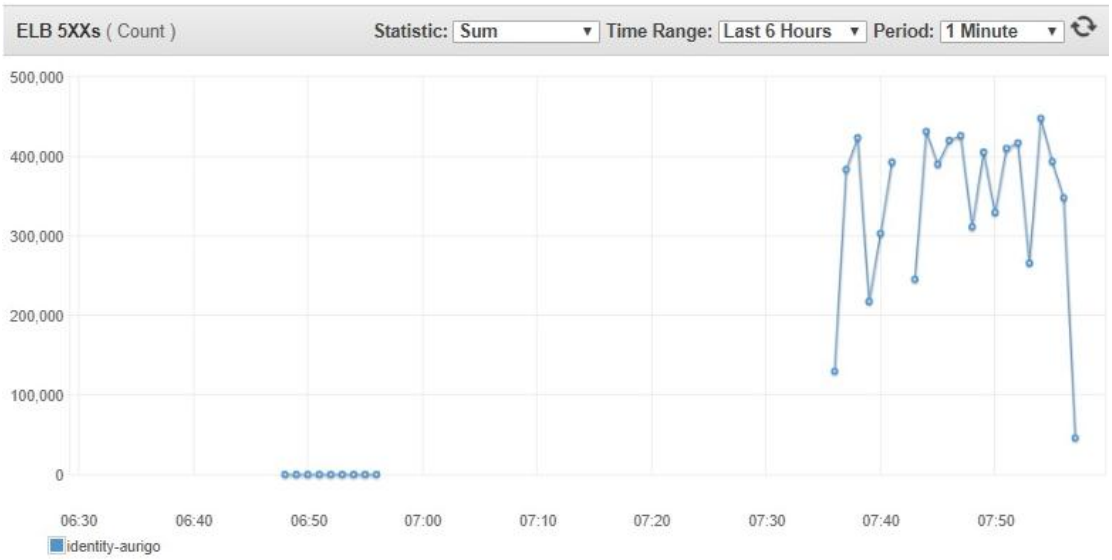

Figure 9: 5xx error graph by load balancer

The threshold time for warming up a new instance was observed to be 5 minutes. Hence it was decided that auto scaling group had to be triggered 5 minutes before the saturation of CPU utilization takes place.1

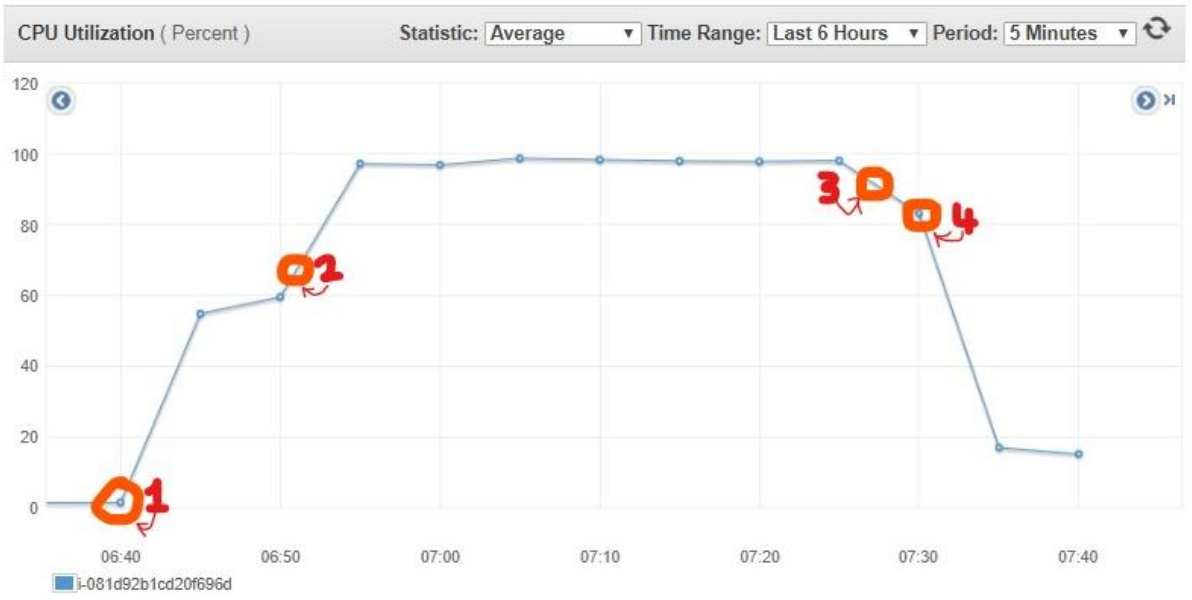

Figure 10: CPU Utilization

To provide an efficient solution to the load balancing issue different parameters can be considered such as CPU utilization, memory down time and utilization. Here a CPU intensive instance "R3.large"[6] is considered. The load is applied at (1) as shown in the Fig.10. The load takes approximately ten minutes to reach the maximum capacity. At point (2) is the threshold of 65% of maximum CPU utilization of the running instance where the second instance will be initiated as a part of auto scaling. But the graph in figure displays the CPU utilization of the running instance, so the graph doesn't depict the autoscaling and load balancing. At point (3) the load is removed, and the utilization starts decreasing, at point (4) 80% utilization the instance is terminated hence cost is reduced [8]. This whole process is implemented, and results are obtained as shown in the fig.10

The number of 5xx errors which signifies that the webpage or the HTTP is incapable of performing the request. It also shows the type of error, if it is a temporary error or a permanent error. The errors were observed to be minimal, and the requests were being served. In the below Fig.11 there is no saturation in CPU utilization due to which the queue length reduced as well as the 5xx errors were reduced drastically when an autoscaling group was implemented with threshold.

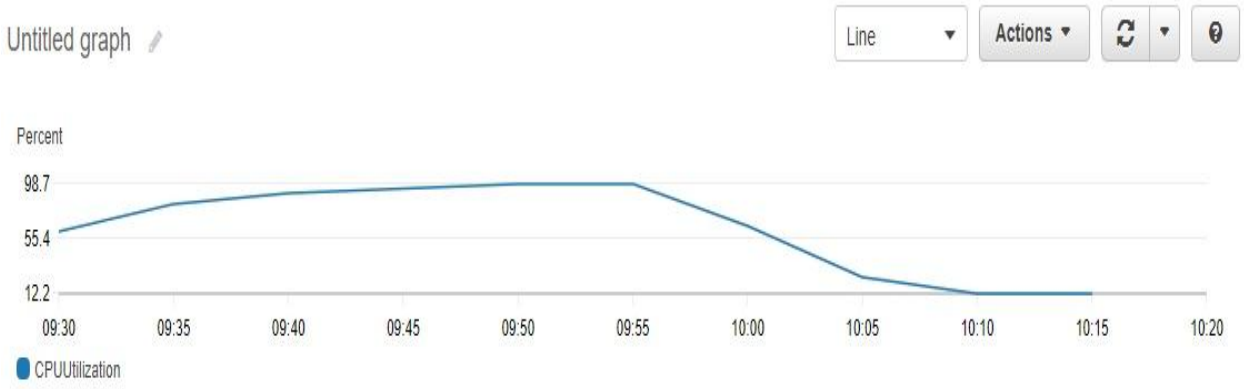

Figure 11: CPU utilization after auto-scaling

4. CONCLUSION

A highly available, fault tolerant infrastructure was achieved with the implementation of autoscaling. The AWS autoscaling group is a powerful tool to have a highly reliable and highly available infrastructure for our requirement. One of the principle targets of this work was to simulate an environment where the number of requests surge, due to which new instances had to be spun up which could service the increasing requests. It was observed that the servicing queue length increases as a result of which the latency of the web-page increases. To mitigate this phenomenon, The conclusion was drawn that for a highly scalable infrastructure the instances have to start warming up when the CPU utilization reaches 65 percent and must terminate the instance when the CPU utilization gown below 80 percentage. This work is Significant in an era where cloud computing taking a lead and most of the companies are shifting from on shore computing to cloud based computing. The work can further be continued to reduce the time for the warming up of the instances and hence achieve the same results with same performance.

REFERENCES

[1] Zibin Zheng, Michael R. Lyu, Irwin King "Component Ranking for Fault-Tolerant Cloud Applications", IEEE transactions on services computing, vol. 5, no. 4, october-december 2012

[2] Ifeanyi P. Egwutuoha, Shiping Chen, David Levy, "A Fault Tolerance Framework for High Performance Computing in Cloud", 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

[3] Lei Yu, Liuhua Chen, Zhipeng Cai, Haiying Shen, Yi Liang, Yi Pan, "Stochastic Load Balancing for Virtual Resource Management in Datacenters", IEEE transactions on cloud computing, vol.5, no.7, november 2014

[4] Z. Zhang, Y. Wang, H. Chen M. Kim, J.M. Xu, H. Lei, "A cloud queuing service with strong consistency and high availability", IBM Journal of Research and Development, Vol. 55, No. 6, pp. 10:1- 10:12, 2011.

[5] Chenhao Qu, Rodrigo N. Calherios, Rajkumar Buyya"A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot fixing", journal of network and computer application, 2016 167-180

[6] Gurudatt Kulkarni, Ramesh Sutar, Jayant Gambhir "cloud computing-infrastructure as serviceamazon ec2", International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622

[7] Richa Thakur, "Auto Scaling Load Balancing Features in Cloud", International Journal of Engineering Trends and Applications (IJETA) – Volume 3 Issue 1, Jan-Feb 2016

[8] Ming Mao, Jie Li, Marty Humphrey, "Cloud Auto-scaling with Deadline and Budget Constraints", 11th IEEE/ACM International Conference on Grid Computing.

[9] Ab Rashid Dar, Dr.D.Ravindran, "Survey On Scalability In Cloud Environment", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5, Issue 7, July 2016.

[10] Gopal Dhaker, Dr. Savita Shiwani, "Auto-Scaling, Load Balancing and Monitoring As service in public cloud", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 4, Ver. I, 2014.

[11] AWS documentation.