1.2 Problem Statement

Since for the web application especially for the service apps like Netflix, Amazon, Flipkart etc. The number of people accessing the application can be tremendous and changes during the time of special sale offer by the company and can become low after the sale ends. The frequent increase for the request of the application can be a bottleneck leads to temporary application shut down for some users. The key issues are load balancing and the scalability of the application. We need to auto scale the application which can manage the request without a problem and if the request for the application is low the auto scale can reduce the resources for the specific application. In this project we aim to develop a web application with the scalability and load balancing using the AWS EC2. We implemented complete application which can auto scale and can efficiently manage the load even if many users access our banking application at the same time.

2. Related Works

2.1 Literature Survey

1) Load-balancing algorithms in cloud computing, Authors and years : EinollahJafarnejadGhomia , Amir Masoud Rahmania, , NooruldeenNasihQaderb,2017. Proposed Methodology is to task scheduling and load-balancing algorithms and present a new classification of such algorithms, for example, Hadoop MapReduce load balancing. Advantages are To avoids the situation in which some nodes become overloaded while the others are idle or have little work to do. Limitations/challenges are Irrespective of mor techniques used balancing of workload among cloud not done efficiently.

2) Elastic Load Balancing for Dynamic Virtual Machine Reconfiguration Based on Vertical and Horizontal Scaling. Autor and Year : Stelios Sotiriadis , Nik Bessis, Senior Member, Cristiana Amza, and Rajkumar Buyya ,2019. Proposed Methodology is ICLB-Inter -cloud load balancer is used that allows scaling up or down the virtual resources (thus providing automatized elasticity), by eliminating service downtimes and communication failures. Advantages are Focuses on cloud load balancing based on dynamic virtual machine reconfiguration when variations on load or on user requests volume are observed. Needed to improve much more elasticity especially in IAAS.

3) Adaptive Resource Allocation for Load Balancing in Cloud. Somnath Mazumdar, Alberto Scionti, and Anoop S. Kumar,2019. ARIMA has been used for forecasting the future application workload behaviour in Cloud. IaaS layer provides more control for balancing the workload. Optimising the PSO for better convergence speed and quality of the solution should be improved.

4) LOAD BALANCING IN AUTO SCALING-ENABLED CLOUD ENVIRONMENTS, Nguyen Hong Son,Nguyen KhacChien,2017, how auto scaler supplementing VMs to subscribed cloud system with round robin load balancer and with active monitoring load balancer. The results of running with round robin load balancer. BATS-Which satisfies the limitation of budget while minimizes service delay, The load balancer works with a fixed number of machines in a subscribed system.

5) RSA and SFQ based Secure Heuristic Load Balancing Approach for Cloud Data Centers, Sweta Dey, Kiran preetkaur, Mehak, Upinder Kaur,2019. Input: Incoming loads Output: Best possible secure assignment of incoming loads to its appropriate VMs, A new efficient safe and secure technique for balancing the dynamic incoming loads in the cloud data centers namely: RSA and SFQ based Secure Heuristic Load Balancing Approach for Cloud Data Centers. HBBLB Algorithm can't provide an efficient secure load balancing approach although it is also based on heuristic rules.

6) An Advanced Algorithm for Load Balancing in Cloud Computing using MEMA Technique, Saher Manaseer, MetibAlzghoul, Mazen Mohmad,2019, Depending on the idea of weighted round robin (WRR) they divided into two parts, the first one is determined by priority messages where the original balancer is divided into normal request balancer and priority request balancer, MEMA technique improves the ability to balance the load among servers (nodes) with more fairness requirements, Discarded request could make the server less reliable with less response time for argent request.

7) Elastic Load Balancing Using Self-Adaptive Replication Management, JUI-PING YANG 2017,Dynamic scheduling with SARM- is robust under a variety of load conditions while maintaining approximately optimal performance without frequently adjusting configuration parameters. It is very easy to implement as efficiently balance the loads among nodes when compared with existing load-balancing schemes with varying numbers of nodes, hotspot data loads, and request patterns. SARM uses active storage management by reserving certain storage space for permanent replicas. Once the amount of temporary and permanent replicas exceeds a pre-defined threshold, the node will reject any new temporary replica assignment.

8) Suboptimal Mechanism For Load Balancing In Cloud Environment, Shikha Pandey
Ashok Kumar Upadhaya,2017. The cloud workload network is constructed based on-Greedy heuristics which has the property to build a incremental diverse set , selecting one compute node at a time in order to maximize the diversity. The technique improves path sharing at time of migration during load balance process. For proper and optimal utilization of resources, distribution of resources so that no overloading occurs at any machine of a cloud network is an important concern.

9) A study on performance measures for auto-scaling CPU-intensive containerized applications, Emiliano Casalicchio,2019. This paper investigate the problem of selecting the most appropriate performance measure to activate auto-scaling actions aiming at guaranteeing QoS constraints. The Kubernetes' Horizontal Pods Auto-scaling(KHPA) algorithm is used to trigger the scaling actions and to estimate the number of containers to be deployed to keep the resource utilization below a specified threshold value. KHPA only relies on absolute-relative CPU utilization correlation model to predict the absolute metric values and to determine, more accurately, the number of containers to be deployed.

10) Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system, NAVTEJ SINGH GHUMMAN, RAJWINDER KAUR, 2017, Improved Max-Min algorithm and Ant colony approach**,** It helps in manage the total makespan (number of unexecuted tasks). For large tasks can have highest priority and smaller tasks can have lower priority.Works only in two phases.Should calculate maximum completion time to a resources in first phase.Assign max completion time with min execution time to resources individually.

11) CLB: A novel load balancing architecture and algorithm for cloud services, Shang-Liang Chen, Yun-Yao Chen ,Suang-Hong Kuo, 2017, A server load balancing method is used and it divided into two types of hardware and software. Efficiency: improve the efficient use of the server and network bandwidth. Reliability and Safety: improve the reliability and security of the server.Scalability: improve the scalability of services,polling approach (also called the Round Robin approach).

12) Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Mohit Kumar , S.C. Sharma, 2017, improving the backfill algorithm (IBA) using balanced spiral.Static algotithms: **First come first serve (FCFS) and shortest job first (SJF).**This algorithm decreases the makespan time. Increases the task to meet the deadline ratio compared with the min-min, the first come-first-serve and the shortest-job-first algorithms in all conditions. Limitation of this paper is load balancer unable to maintain the session in multi-tenant environment when same user request the multiple VM instances.

13) Load balancing in cloud computing: Challenges & issues, Ms. Shalini Joshi, **Dr. Uma Kumari, 2017,** Static and dynamic load balancing algorithms., Works well with number of processes that is larger than number of processor. Round Robin does not demand for inter-process communication. High degree of interprocess communication which leads to bottleneck state. It can take the condition of starvation.

14) **Dynamic and elasticity ACO load balancing algorithm for cloud computing. M.Padmavathi, Prof.Shaik. Mahaboob Basha, 2017, ANT COLONY OPTIMIZATION (ACO) LOAD BALANCING ALGORITHM,** Performs well with optimal number of resources. **Load scheduler makes load balancing decisions that depends on the system load information.** Task scheduling approaches don't give the guarantee of load balancing. As the system size increasing, it does not increase throughput.

15) Energy Efficient Dynamic Threshold Based Load Balancing Technique in Cloud Computing Environment. Shivani Gupta, Damodar Tiwari, Shailendra Singh, 2017, Dynamic threshold based approach and an energy efficient VM placement approach for the cloud., increased resource utilization ratio which further leads to enhancing the overall performance thereby achieving maximum client satisfaction, The techniques which are proposed in existing work require extra steps to balance network load. The technique needs to be proposed which does not require extra steps to balance network load.

16) A Comparative Study on Load Balancing and Energy Efficiency Techniques in Cloud Paradigm. Shivani Bajaj, Manisha Malhotra, 2017, Round robin, ant colony and opportunistic load balancing. Performs good with high utilized resources. Utilizing the increased system resources to increasing throughput. Only utilized in Complex networks. Low scalability and Performance.

17) Survey on Various Load Balancing Techniques in Cloud Computing. Ragesh Raju, Sathyendra Bhat J, Bikramjitathokpam, Rio G.L. D'Souza, Throttled Load Balancing Algorithm (TLB), Load Balance Improved Min-Min Scheduling Algorithm (LBIMM), Min-Min Scheduling Algorithm. It allows workloads and computing resources to be better distributed across several servers, either on-premise or in the cloud. Develop resource allocation model for fine grained time specific workloads,Architecture will be applied to dynamic cloud provisioning

18) Analysis of load balancing in cloud data centers. Sweekriti M. Shetty, · Sudheer Shetty, A Multi-Class Load Balancing Algorithm (MCLB), This can better utilize your system and improve its overall response time either on premise or on the cloud. Extend the scheme to more realistic data center architecture,Authors plan to adopt cluster-based paradigm in future design

19) Dynamic Load Balancing Approach for Minimizing the Response Time Using An Enhanced Throttled Load Balancer in Cloud Computing- 2019. G. JustyMirobi,L. Arockiam. The authors used a dynamic load balancing approach using enhanced Throttled Load balancer methodology to enhance and develop the load balancer for allocating work evenly to all the virtual machines and thus migrating the overloaded work from one virtual machine to another underloaded virtual machine. The dynamic load balancing algorithm is based on the progress of the cloud environment. Reduced delay time and response time of the service. Deliver better performance than static load balancing and offer run time changes capability.

20) Performance Modeling and Verification of Load Balancing in Cloud Systems Using Formal Methods- 2019, Shenghui Chen, Zhiming Fan,Haiying Shen, Lu Feng,By using various algorithms to check the quantitative reliability the authors built the virtual machine migration model and built an MDP based performance model in PRISM. With this model the authors demonstrated that we can checkthe range of properties elaborated in probabilistic temporal logic regarding the performance of thealgorithm. With this quantitative verification can be done with the safety, liveness, critical indicatorand multi-objective properties to get insight about performance of the cloud.The insight can be helpful to cloudservice providers and managers to better increase the performance.

2.2 **Comparative statement** (10 latest Journal papers in the current domain)

1.	An energy efficient load balancing on cloud computing using adaptive cat swarm optimization	K. Balaji, P. Sai Kiran, M. Sunil Kumar 2020	They developed a load balancing system using adaptive cat swarm optimisation (ACSO) algorithm. The efficiency of the proposed technique is analysed by using different value metrics and performance differentiated with other method	It will help to deploy data centres(DC) across the globe that includes thousands of servers. They used the tool called Cloudism to test performance of their algorithm	Their methodology does not capture any real-time load balancing data ,as they used the Cloudsim tool which is a simulation tool not a actual software.
2	Load balancing in cloud computing: A big picture	Sambit Kumar Mishra ,Bibhudatta Sahoo , Priti Paramita Parida 2020	The load balancing in cloud computing with a multi-objective system is a well known NP-complete problem	This papers compares with many algorithms and shows which is best	Evaluating the proposed algorithms in a real-world cloud deployment is not done
3	Elastic Technique for Load Balancing in Cloud Computing	Sovban Nisar, Deepika Arora, Navneet Verma 2020	Elastic use of Cloud Computing users are able to share processing power, storage space, bandwidth, memory and software. As if somebody is using Cloud computing then their	This helps user to spend only less cost as they must pay based on usage.	No specific approach, any tool or implementation not is not specified properly.

			resources get shared along with that the cost is also getting shared.		
4	Stochastic Load Balancing for Virtual Resource Management in Datacenters	Lei Yu, Zhipeng Cai Yi Liang 2020	The workload uncertainty of VMs with a real world trace, formally define the problem of stochastic load balancing using VM migration, and	The previous load balancing schemes detect hotspots and decide VM migrations based on deterministic resource demand prediction, which can lead to poor performance and severe SLA violations Stochastic load blancing leads to better performance	VM migration algorithm considers the network topology by taking into account the distance from the source to the destination PM for a VM migration, it does not consider the bandwidth usages on the links of the migration path
5	Wisely Randomized Weighted Throttled Load-Balancing Algorithm for Cloud	Mrs. Urvashi Sapra and Dr. Anoop Sharma, 2020	Wisely, Randomized Weighted Throttled Algorithm (WRWTA)	The event of two indistinguishable machines simultaneously will give the above favourable position extra alluring computational time.	Only utilized in Complex networks. Low scalability and Performance.
6	CLOUD LOAD BALANCING AND DATA SHARING APPROACH OVER MULTIPLE VIRTUAL MACHINE	HabeebullahHussaini Syed, TriptiShrivastava , Raj Kumar Paul, 2021	Algorithm based on the enhanced version of SVM for load balancing approach over the cloud data centre.	Control heterogeneity. It is adaptive to dynamic environments. Outstanding in case of fault tolerance.	It is not based on distributed balancing approach. In round robin, there are no expectations to obtain better performance.

7	Improved Hybrid Algorithm Approach based Load Balancing Technique in Cloud Computing	Srinivas Rao Gundu &T.Anuradha, 2019	Round Robin algorithm and combination of round robin and throttled algorithm.	A slight improvement is observed in makespan waiting time and burst time. Equally spread current execution algorithm can be combined with hybridization concept.	The additional load on the scheduler to decide the size of quantum. It works properly only if all virtual machines in data centre have the same hardware configuration.
8	**Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing**	Mohit Kumar, Kalka Dubey, S.C. Sharma, 2018	A load balancing algorithm considering QoS as parameter.	Load scheduler makes load balancing decisions that depends on the system load information.	Reduces as system diversity increases. Degrades as population diversity increases.
9	Creating Elasticity with Enhanced Weighted Optimization Load Balancing Algorithm in Cloud Computing	Neha Tyagi , Ajay Rana, Vineet Kansal, 2019	A Mixture of weighted-optimization and enhanced weighted optimization algorithm.	Round Robin does not demand for inter-process communication. A number of local process allocations are done.	In round robin, there are no expectations to obtain better performance. Some important constraint is not considered like cost, heterogeneous virtual machine
10	Star Hotel Hospitality Load Balancing Technique in Cloud Computing Environment	V. Sakthivelmurugan , R. Vimala and K. R. Aravind Britto, 2019	Star Hotel Load Balancing Method (SHLB) algorithm	It reduce the execution time of user's application. It lowers the makespan and minimize task execution time.	The existing techniques for load balancing require hardware and software which increase system complexity. The technique is required which does not need any

					hardware and software to network load.

2.3 Hardware Requirements

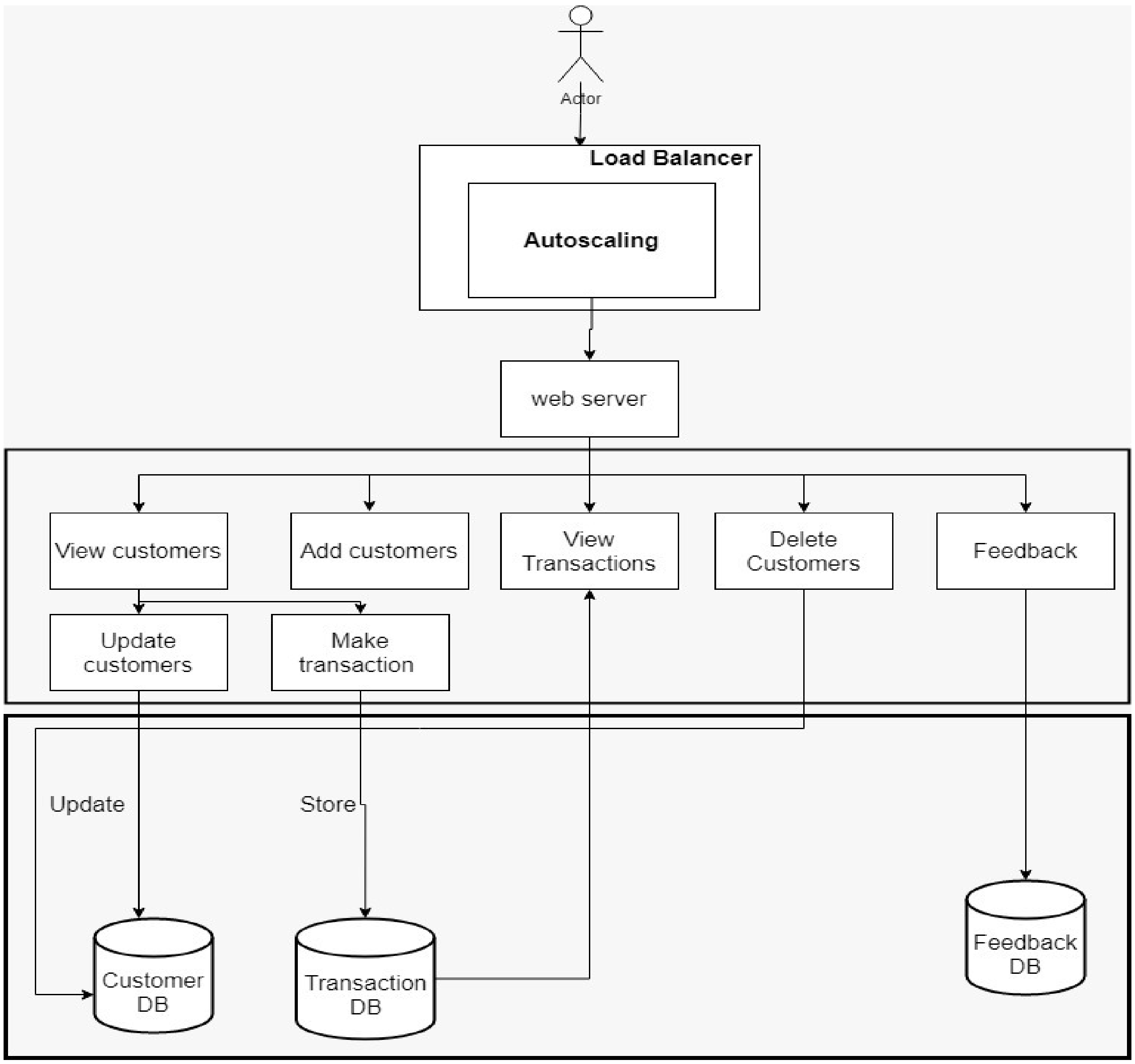
Specs	Minimum Specification	Recommended Specifications
Processor	Intel Core i3 or equivalent	Intel Core i5 or better*
Memory	4 GB	8 GB or more
Hard Drive	80 GB hard drive space	120 GB hard drive spaceor larger

2.4Software Requirements

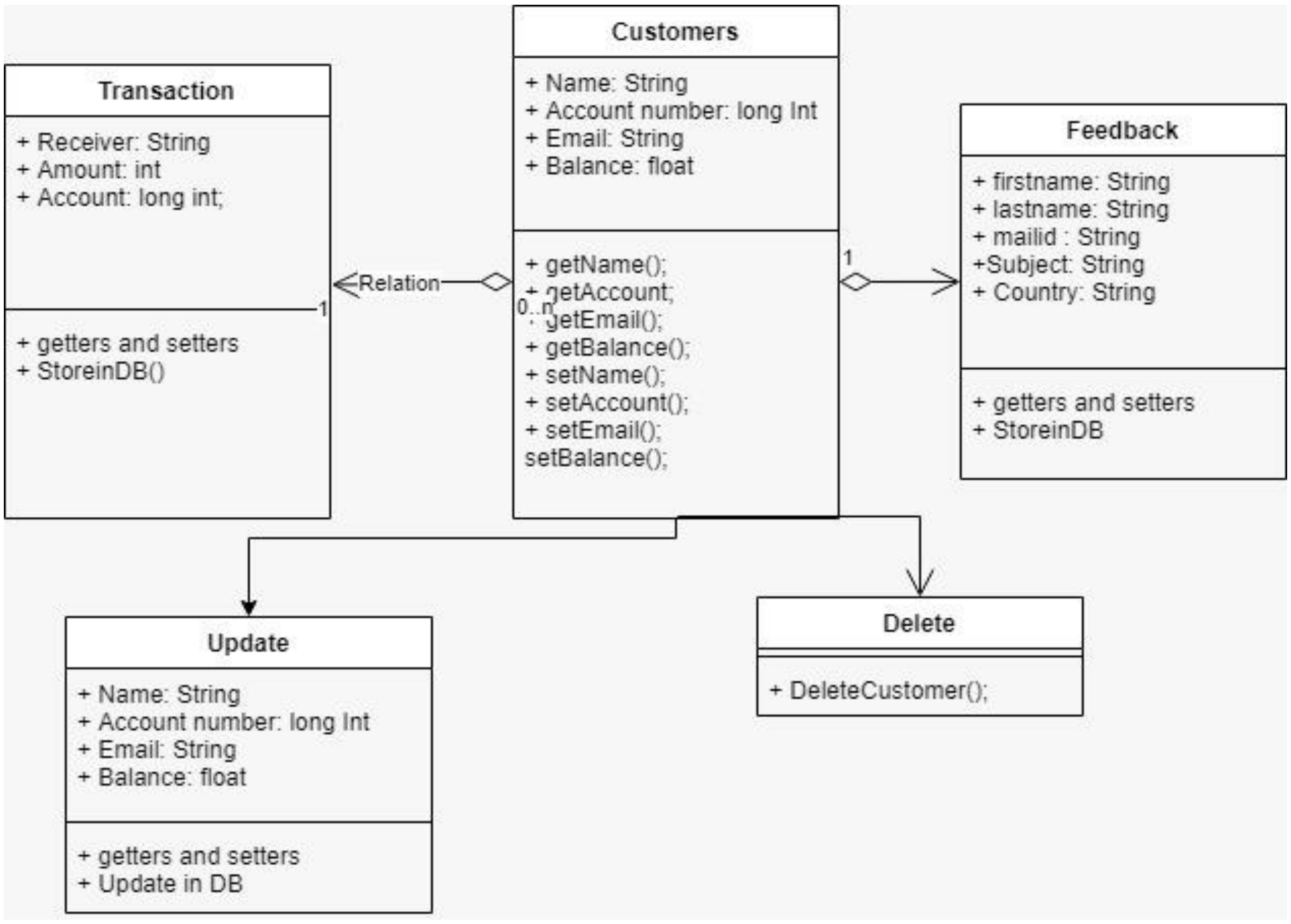
1) Java Spring boot

3. System Design

3.1 High-Level Design (Black Box design)



3.2 Low-Level Design(Detailed design)

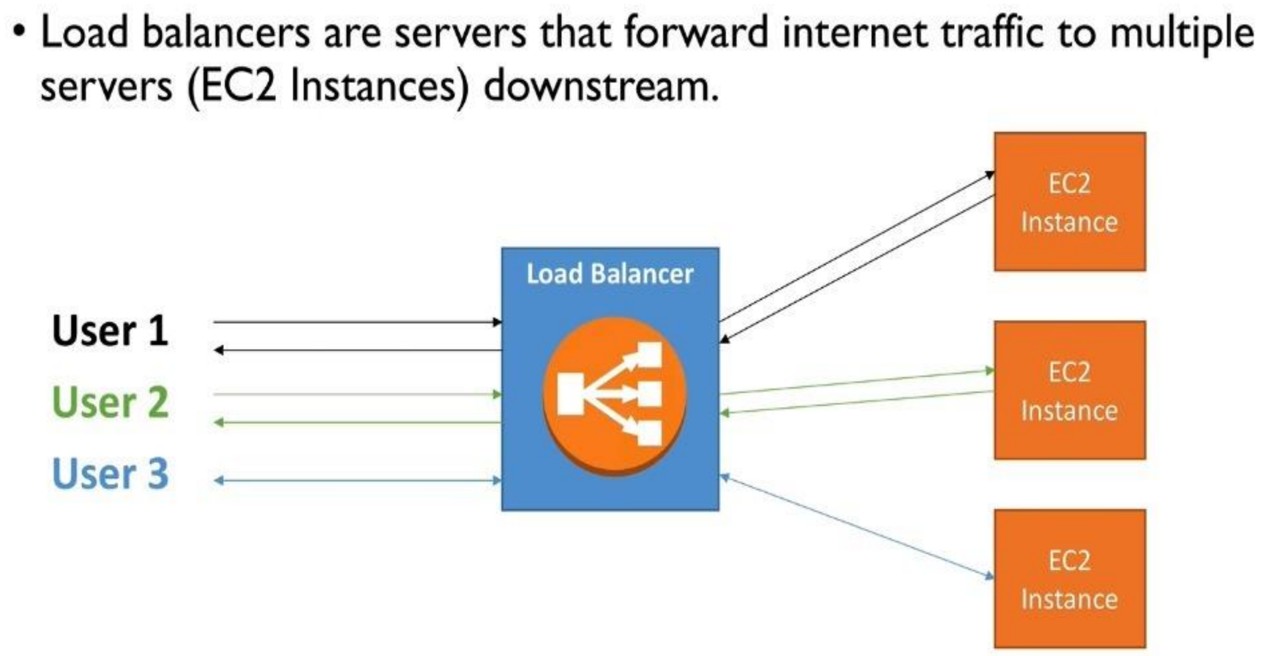


4. System Implementation

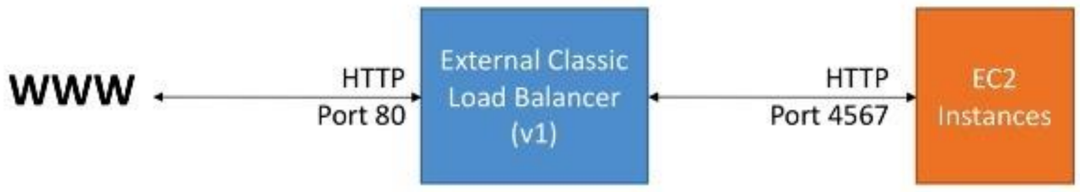
4.1 Algorithms:

The Elastic load balancer accepts the internet traffic from clients/users and send request from the registered EC2 instances, in one or more availability zones. When a load balancer detects unhealth EC2 instance because of any natural disaster or like that, it stops routing to that specific EC2 instance which is unhealth and resumes the routing to that target when it detects the target is healthy again after the instance getting fixed, recovered from disaster or configured properly.

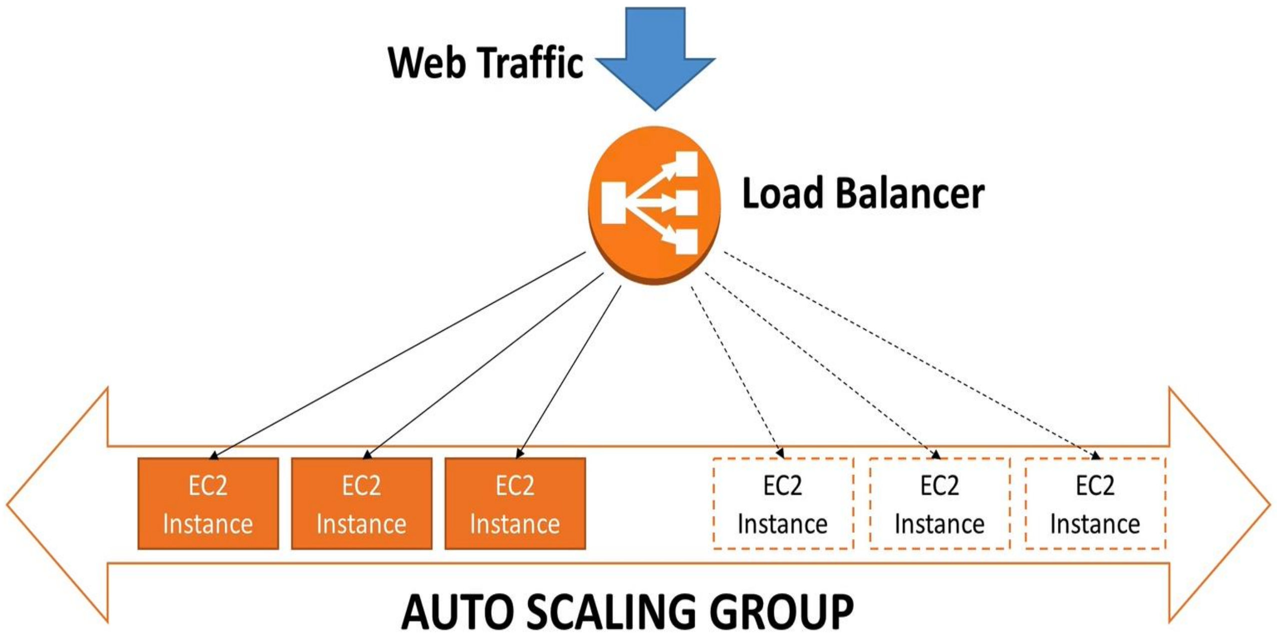
Load Balancing Algorithm:

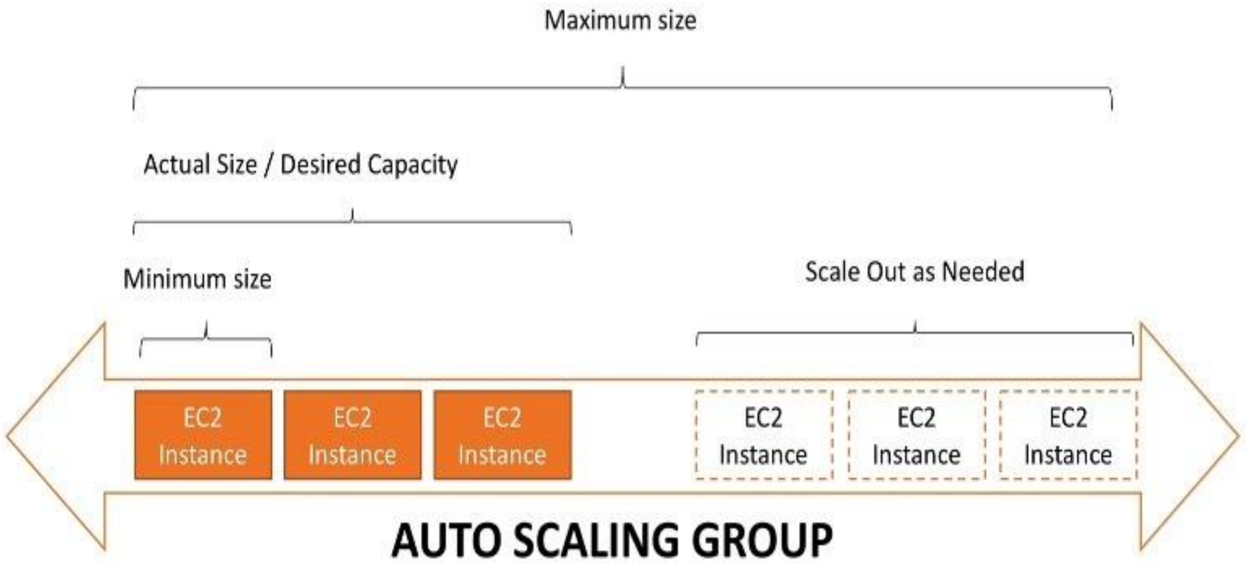


- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



Auto Scaling Algorithm:





4.2 Module Development:

BankingApplication.java code:

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BankingApplication {

	public static void main(String[] args) {
		SpringApplication.run(BankingApplication.class, args);
	}

}
```

BankingController.java file:

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class BankingController {

	String sender;
	String mail;

	@Autowired
	private CustomerService custser;

	@Autowired
	private Customerrepository rep;

	@Autowired
	private CustomerService customerservice;
```

```
@Autowired
private TransactionService serv;

@Autowired
Transactionrepo trpo;

@Autowired
Feedbackrep feedrep;

private Transaction tr;


@GetMapping("/")
public String home() {
        return "index";
}
@PostMapping("/customersubmit")
public String customer(Customers cust) {
        rep.save(cust);
        return "index";
}

@PostMapping("/Transfersubmit")
public String transfer(Transaction trans,Customers cus) {
        Customers some=rep.findByEmail(mail);
        some.setBalance(some.getBalance()-trans.getAmount());
        trans.setSender(sender);
        trpo.save(trans);
        rep.save(some);
        return "Transactions";
}

@GetMapping("/Customers")
public String homies(Model model) {
        model.addAttribute("customers",custser.getAllCustomers());
        return "Customers";
}

@GetMapping("/Transactions")
public String homies2(Model model) {
        model.addAttribute("transaction",serv.getAllTransaction());
        return "Transactions";
}
```

```
	@GetMapping("/add")
	public String add() {
		return "add";
	}

	@GetMapping("/Trans")
	public String addo() {
		return "Transfer";
	}


	@GetMapping("/showFormForUpdate/{email}")
	public String showFormforupdates(@PathVariable(value="email") String email,Model model) {
		Customers customer= customerservice.getCustomersById(email);
		model.addAttribute("customers",customer);
		return "update";
	}

	@GetMapping("/showFormFortrans/{email}")
	public String showFormfortrans(@PathVariable(value="email") String email,Model 
model,Transaction tra) {
		Customers customer= customerservice.getCustomersById(email);
		Customers some=rep.findByEmail(email);
		mail=email;
		//Transaction tx=trpo.getById(idd);
		sender=customer.getName();
		Customers nn=rep.findByName(sender);
		tra.setSender(sender);
		//trpo.save(tra);
		some.setBalance(some.getBalance()-tra.getAmount());
		rep.save(some);
		model.addAttribute("customers",customer);
		return "Transfer";
	}

	@GetMapping("/Feedback")
	public String homieez() {
		return "Feedback";
	}

	@PostMapping("/feedbacksubmit")
	public String greetsings(Feedback feedback) {
		feedrep.save(feedback);
		return "index";
	}
	@GetMapping("/Deletecustomer/{email}")
	public String deleter(@PathVariable(value="email") String email,Model model) {
		this.customerservice.deleteCustomerByEmail(email);
		return "index";
	}


	@GetMapping("/Delete")
	public String delete(Model model) {
		model.addAttribute("customers",custser.getAllCustomers());
		return "Delete";
	}
}
```

Customers.java

```
package com.example.demo;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Customers {

	public Customers() {

	}

	@Id
	private String email;

	private String name;

	private long balance;

	private int accountnumber;

	public int getAccountnumber() {
		return accountnumber;
	}


	public void setAccountnumber(int accountnumber) {
		this.accountnumber = accountnumber;
	}

	public Customers(String email, String name, long balance,int accountnumber) {
		super();
		this.email = email;
		this.name = name;
		this.balance = balance;
		this.accountnumber=accountnumber;
	}


	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public long getBalance() {
		return balance;
	}

	public void setBalance(long balance) {
		this.balance = balance;
	}


}
```

Feedback.java

```
package com.example.demo;
import javax.persistence.Entity;
import javax.persistence.Id;

import org.springframework.beans.factory.annotation.Value;

@Entity
public class Feedback{
	@Id
	@Value("")
	 private String mailid;

		public Feedback(String mailid, String firstname, String lastname, String country, String
subject) {
		super();
		this.mailid = mailid;
		this.firstname = firstname;
		this.lastname = lastname;
		this.country = country;
		this.subject = subject;
	}
		private String firstname;

		  private String lastname;

		  private String country;

		  private String subject;

	  public String getMailid() {
		return mailid;
	}
	public void setMailid(String mailid) {
		this.mailid = mailid;
	}
	public String getFirstname() {
		return firstname;
	}
	public void setFirstname(String firstname) {
		this.firstname = firstname;
	}
	public String getLastname() {
		return lastname;
	}
	public void setLastname(String lastname) {
		this.lastname = lastname;
	}
	public String getCountry() {
		return country;
	}
	public void setCountry(String country) {
		this.country = country;
	}
	public String getSubject() {
```

```
                return subject;
        }
        public void setSubject(String subject) {
                this.subject = subject;
        }

public Feedback() {

}
}
```

Transaction.java File:

```
package com.example.demo;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity
public class Transaction {

        public Transaction() {

        }

@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private long Id;


private long amount;

private int accountnumber;

private String sender;

private String receiver;

public Transaction(long id, long amount, int accountnumber, String sender, String receiver) {
        super();
        Id = id;

        this.amount = amount;
        this.accountnumber = accountnumber;
```

```
		this.sender = sender;
		this.receiver = receiver;
	}

	public long getId() {
		return Id;
	}

	public void setId(long id) {
		Id = id;
	}

	public long getAmount() {
		return amount;
	}

	public void setAmount(long amount) {
		this.amount = amount;
	}

	public int getAccountnumber() {
		return accountnumber;
	}

	public void setAccountnumber(int accountnumber) {
		this.accountnumber = accountnumber;
	}

	public String getSender() {
		return sender;
	}

	public void setSender(String sender) {
		this.sender = sender;
	}

	public String getReceiver() {
		return receiver;
	}

	public void setReceiver(String receiver) {
		this.receiver = receiver;
	}


}
```

4.3 Test Cases

Using the tool called Restful Stress we stressed our banking application:

Using this with the help of CloudWatch metrics we can stress this banking application to stressful state such as CPU, NetworkIn-bytes which will attend peak load, by using CloudWatch alarm to notify the auto scaling group to scale-in or scale-out accordingly. So that app will face low down time.

5. Results and Discussion

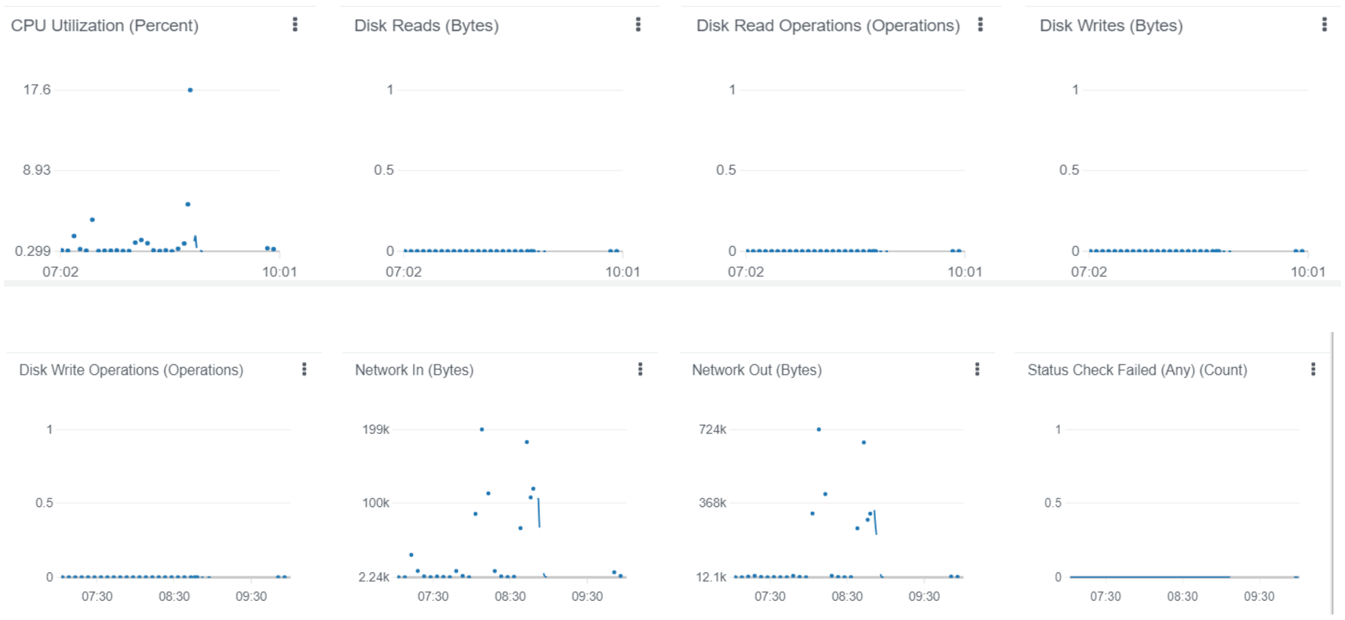
5.1 Implementation Results

After Review 2 we had full stack bank like application. Now the application is load balanced and auto scaled. The instances can be added and removed dynamically based on the alarm set in CloudWatch. The Auto scaling group watch for the trigger and auto scale in or scale out based on the load. The load balancer will direct the traffic to every instances or by using the ASG group the instances will be initialized to match the desired capacity. The instances created will automatically register and de-register the instances. The security check will be happening in the load balancer itself. If the request is not in the rule. The Load balancer itself won't serve the response and the request is not even processed by EC2 if the request is out of the inbound rules.

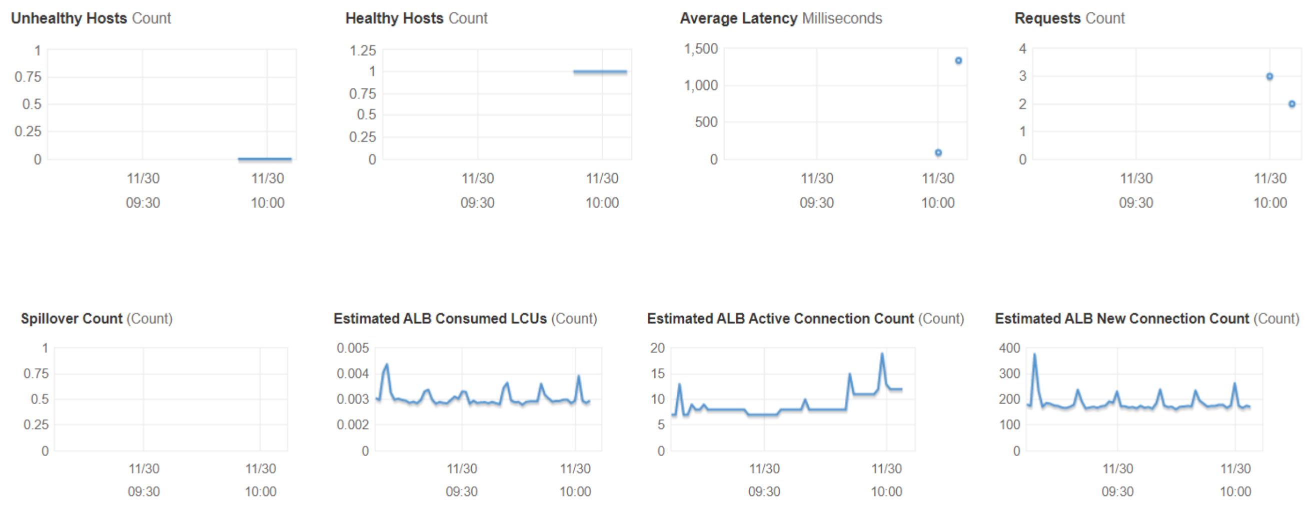
5.2 Metrics

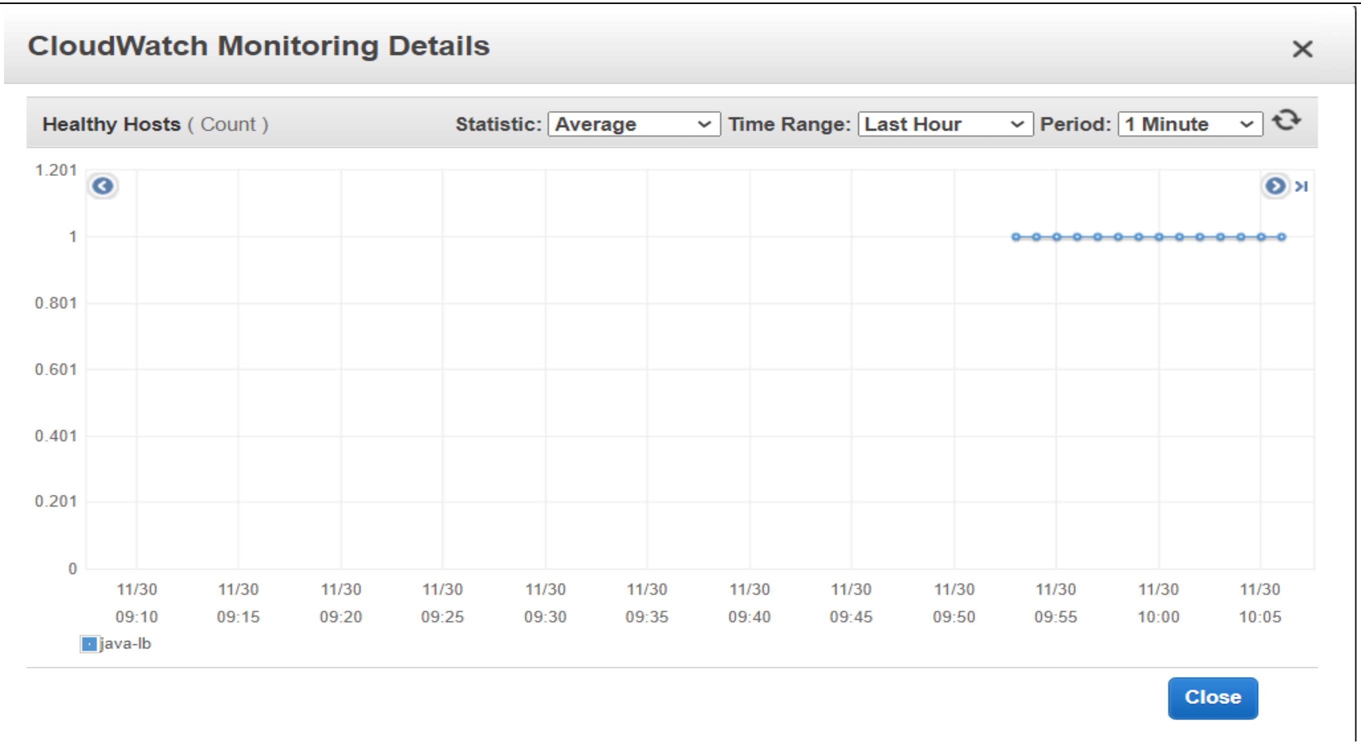
We had two metrics to load trust our application. At first we used CPU Utilization percentage as a metric and later we used NetworkIn(BYTES) metrics for the test. When the bytes reached 100k above the alarm will be triggered and to serve the app without delay the Auto scaling group will add the two extra instances to maintain the balance if the NetworkIn bytes is stayed above 100k for the one consecutive 300 seconds and once the load is reduced and if the NetworkIn bytes came below 50k the auto scale group will remove 1 instances if the bytes stayed for 1 consecutive 300 seconds.

Metrics on Running EC2 instances.



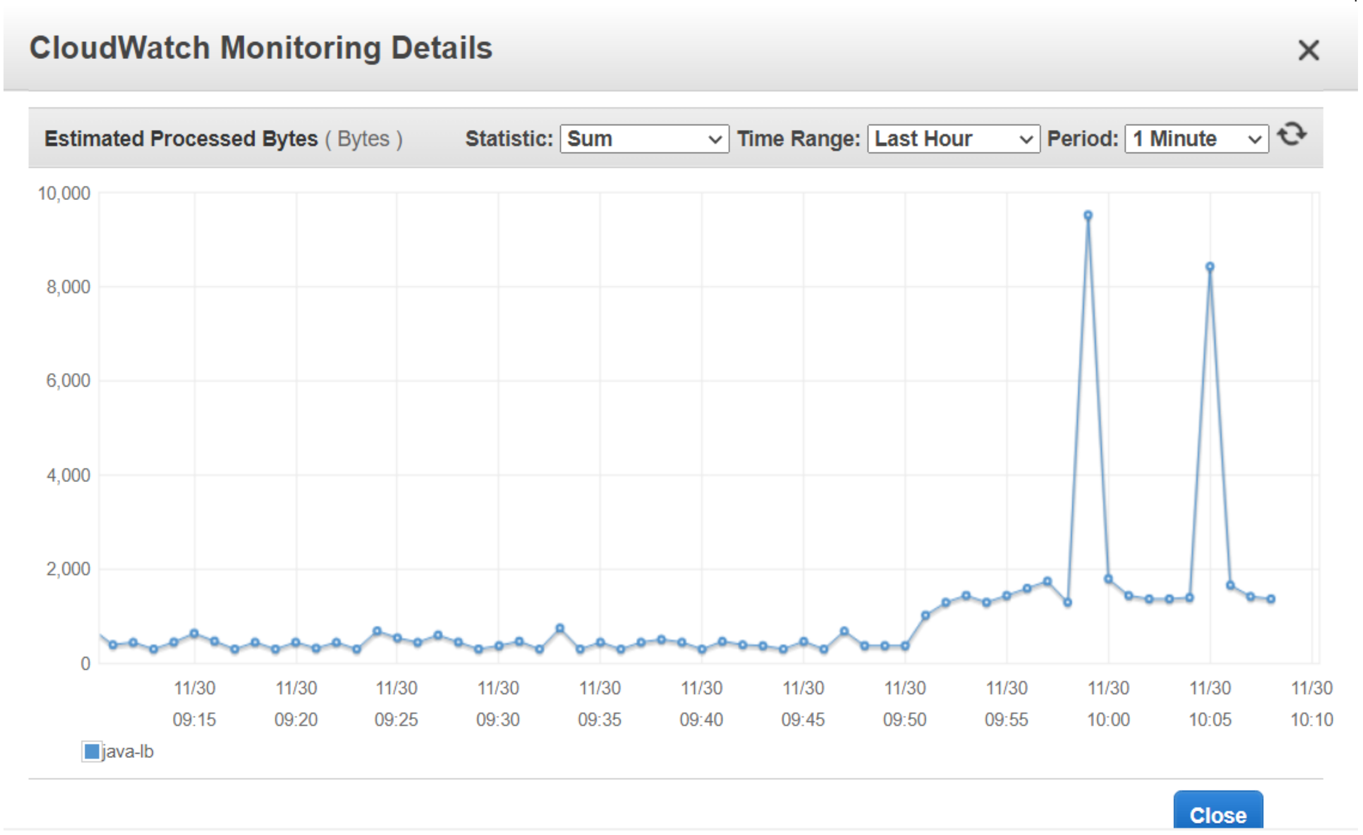
Metrics on activating ELB





CloudWatch Monitoring Details

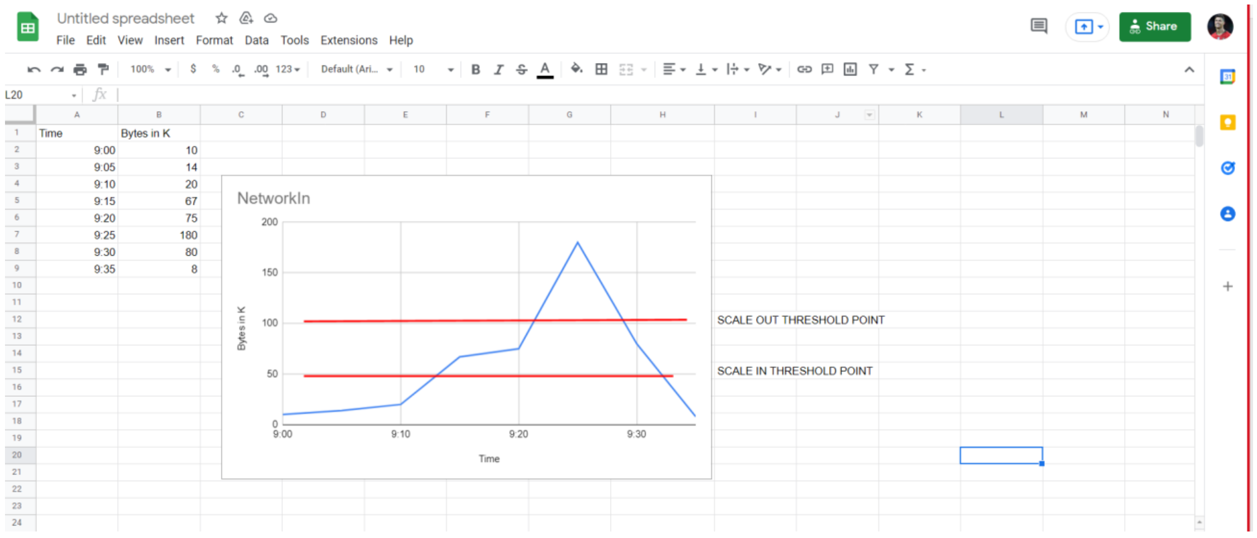
Healthy Hosts (Count) Statistic: Average Time Range: Last Hour Period: 1 Minute



CloudWatch Monitoring Details

Estimated Processed Bytes (Bytes) Statistic: Sum Time Range: Last Hour Period: 1 Minute

5.3 Results in table/Graph/Data:


Untitled spreadsheet
File Edit View Insert Format Data Tools Extensions Help
Share
L20
NetworkIn
Bytes in K
Time
SCALE OUT THRESHOLD POINT
SCALE IN THRESHOLD POINT

Time	Bytes in K
9:00	10
9:05	14
9:10	20
9:15	67
9:20	75
9:25	180
9:30	80
9:35	8

Document Link:
https://docs.google.com/spreadsheets/d/15dOTkdvaZl6TkPkapc6VSWzYJC1MLbvwt_9TImYtHdo/edit?usp=sharing

5.4 Mapping the results with problem statement and existing systems

Companies like, Flipkart, Amazon during the Big Billion Sale or Amazon great Indian freedom sale. The number of users gets significantly increased when compare to other months. So during this time the application becomes very slow or it take so much time to checkout the products this will happen when we don't have enough EC2 instances to run the application or Elastic load balancing and auto scaling is not included. After including the ELB and Auto Scaling our banking application is very stable even after applying the Stress Test. Before that our EC2 instances became unhealthy because of overloading. So the results we conclude is ELB and Auto Scaling play a vital role in handling a large number of users.

6. Conclusion and Future Developments

We successfully implemented the java application with Auto scaling and Load Balancing which can withstand the load with the help of Auto Scaling group. The load is evenly spread across multiple availability zones (AZ's)

Future Work:

We plan to develop the app more interactive and also to enable more features on the way!

7. References

1) Vishalika and D. Malhotra, "LD_ASG: Load Balancing Algorithm in Cloud Computing," *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2018, pp. 387-392, doi: 10.1109/PDGC.2018.8745948.

2) S. Sotiriadis, N. Bessis, C. Amza and R. Buyya, "Elastic Load Balancing for Dynamic Virtual Machine Reconfiguration Based on Vertical and Horizontal Scaling," in *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 319-334, 1 March-April 2019, doi: 10.1109/TSC.2016.2634024.

3) Balaji, K. & Kiran, P. & Kumar, M.. (2021). An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. Materials Today: Proceedings. 10.1016/j.matpr.2020.11.106.

4) Somnath Mazumdar, Alberto Scionti, and Anoop S. Kumar,"Adaptive Resource Allocation for Load Balancing in Cloud ". K.K. Birla Goa Campus, 403726, Goa, India : 2017.

5) Sambit Kumar Mishra ,Bibhudatta Sahoo , Priti Paramita Parida. "Load balancing in cloud computing: A big picture", Journal of King Saud University, 2018

6) Nguyen Hong Son, Nguyen KhacChien, "LOAD BALANCING IN AUTO SCALING- ENABLED CLOUD ENVIRONMENTS", University of the People's Police, Ho Chi Minh City, Viet Nam 2017.

7) Sweta Dey,Kiran preetkaur, Mehak, Upinder Kaur," RSA and SFQ based Secure Heuristic Load Balancing Approach for Cloud Data Centers", Coimbatore, India,2019.

8) Saher Manaseer, MetibAlzghoul, Mazen Mohmad, "An Advanced Algorithm for Load Balancing in Cloud Computing using MEMA Technique", International Journal of Innovative Technology and Exploring Engineering (IJITEE),ISSN: 2278-3075, Volume-8 Issue-3, January 2019.

9) Sovban Nisar, Deepika Arora, Navneet Verma," Elastic Technique for Load Balancing in Cloud Computing", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-6, March 2020.

10) JUI-PIN YANG," Elastic Load Balancing Using Self-Adaptive Replication Management",November 2016.

11) Shikha Pandey, Ashok Kumar Upadhaya, "Suboptimal Mechanism For Load Balancing In Cloud Environment", Chennai, India, 2017.

12) Lei Yu,,Zhipeng Cai,Yi Liang," Stochastic Load Balancing for Virtual Resource Management in Datacenters", IEEE, 2016.

13) Emiliano Casalicchio, "A study on performance measures for auto-scaling CPU-intensive containerized applications", Sapienza University of Rome, 2019.

14) **Navtej Singh Ghumman, Rajwinder Kaur, "Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system",6th ed.Denton, U.S.A : 2015.**

15) Shang-Liang Chen, Yun-Yao Chen∗ ,Suang-Hong Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services", Volume 58. National Cheng Kung University, Tainan : 2016.

16) Mohit Kumar , S.C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment", IIT Roorkee, India : 2017.

17) V. Sakthivelmurugan , R. Vimala and K. R. Aravind Britto, "Star Hotel Hospitality Load Balancing Technique in Cloud Computing Environment", Tamil Nadu, India : 2018.

18) Shalini Joshi, Uma Kumari,"Load balancing in cloud computing: Challenges & issues", Greater Noida, India : 2017.

19) M.Padmavathi, Shaik. MahaboobBasha,"Dynamic and elasticity ACO load balancing algorithm for cloud computing", June 2017.

20) Neha Tyagi; Ajay Rana; Vineet Kansal,"Creating Elasticity with Enhanced Weighted Optimization Load Balancing Algorithm in Cloud Computing",Greater Noida, India 2 Amity University, Noida, India : 2019.

21) Mohit Kumar, Kalka dubay, S.C.Sharma, "Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing", Volume 125. IIT Roorkee, India : 2018.

22) Mrs. Urvashi SapraDr.,Anoop Sharma, " Wisely Randomized Weighted Throttled Load

Balancing Algorithm For Cloud", Volume 5 Issue 7. Singhania University, Rajasthan : 2020.

23) Shivani Gupta, Damodar Tiwari, Shailendra Singh, "Energy Efficient Dynamic Threshold Based Load Balancing Technique in Cloud Computing Environment", Vol. 6 (2).
Rajiv Gandhi ProudyogikiVishwavidyala, Bhopal Madhya Pradesh (India):2015.

24) Shivani Bajaj, Manisha Malhotra**, "**A Comparative Study on Load Balancing and Energy Efficiency Techniques in Cloud Paradigm**",** Chandigarh University, Gharuan, Mohali, India : 2017.

25) Srinivasa Rao Gundu & T. Anuradha, "Improved Hybrid Algorithm Approach based Load Balancing Technique in Cloud Computing",Volume 19 Issue 2 Version 1.0, Dravidian University Kuppam : 2019.

26) HabeebullahHussaini Syed, TriptiShrivastava , Raj Kumar Paul**,** **"**CLOUD LOAD BALANCING AND DATA SHARING APPROACH OVER MULTIPLE VIRTUAL MACHINE", Volume 6 issue 7.Bhopal : 2017

27) Ragesh Raju, Sathyendra Bhat J, Bikramjitathokpam, Rio G.L., "Survey on Various Load Balancing Techniques in Cloud Computing**",**St Joseph Engineering College, Mangaluru : 2017.

28) Sweekriti M. Shetty, Sudheer Shetty**,** **"**Analysis of load balancing in cloud data centers",Sahyadri College of Engineering and Management, Adyar, India : 2019.

29) G. Justy Mirobi,, L. Arockiam," Dynamic Load Balancing Approach for Minimizing the Response Time Using An Enhanced Throttled Load Balancer in Cloud Computing", Tirunelveli, India, 2020.

30) Shenghui Chen,Zhiming Fan,,Haiying Shen, Lu Feng," Performance Modeling and Verification of Load Balancing in Cloud Systems Using Formal Methods", Monterey, CA, USA, 2020.