Optimal Load Balancing Strategy in a Virtual Environment

Sahana B[1], Abhay Deshapande[1*], AV Sruthi[1], Deekshith KN[1], Diksha Dinesh[1]

[1]Department of Electronics and Communication Engineering, RV College of Engineering®, Bengaluru

Abstract

In a virtual environment, load balancing is the method of assigning a collection of tasks through a range of resources, in order to make their total functionality more efficient. Load balancer is used to increase the device efficiency for the user, and overall service stability at the same time. Resource allocation in a virtual environment, in this paper, is accomplished using Amazon Web Services (AWS) as the basis for our virtual environment. Amazon Relational Database Services (RDS) supports high storage capacity and scalability. HAProxy performs the duties of a Load Balancer, by allocating incoming requests amongst the virtual servers. The virtual servers, which are created by setting up Amazon Elastic Computer Cloud (EC2) instances have same properties such as load bearing capacity and reaction time. One of the EC2 instances is configured to host HAProxy as the load balancer. The load bearing potential of HAProxy is inversely proportional to the execution time, ensuring faster processing of the incoming requests. The proposed model is highly efficient, as it is noticed that the number of user increases the execution time varies by small amount. For up to 5 users the average execution time is 0.690 ms whereas for the user range of 20-25 the average execution time is 0.801 ms. For a scenario where the web servers differ in terms of load bearing capacity, the algorithm may be modified by assigning weights. HAProxy load balancer optimizes resource allocation and provides flexibility.

Keywords: *Access Log Files, Amazon Relational Database Services (RDS), Amazon Web Services (AWS), Elastic Computer Cloud (EC2) Instances, HAProxy, Virtual Machines VM.*

1.0 Introduction

Cloud computing is an essential technology in today's continuously technically advancing world. Providing different types of user services through internet form the basis of cloud computing. These services include tools and applications like data storage, servers, databases, networking, etc. So long as an electronic user has access to the internet, they have access to the data and software programs needed to operate it. In cloud computing, virtualization means creating a virtual instance of hardware computer systems, disk tools, and

**Mail address: Abhay Deshpande, Associate Professor, Department of Electronics and Communication Engineering, RV College of Engineering®, Bengaluru – 59*
Email: abhayadeshpande@rvce.edu.in, Ph.: +91 9980815636

network computing services. A well-known example of virtualization is the drive facility offered by Google which helps millions of users to store and share data. Majority of servers used by companies is based on a pay and use policy. Best examples of cloud providers currently are Google cloud, Microsoft Azure, Amazon Web Services (AWS), Alibaba Cloud, IBM cloud, Salesforce, SAP, etc. The conventional methods of maintaining a server is highly inefficient compared to today's virtual cloud servers. Maintaining large number of physical servers is costly; moreover, not all the servers are used at every point of time. Server usage depends on sent requests. Excessive number of requests leads to overloading of a server, hence causing it to shut down. Using cloud servers overcomes all these problems as large number of servers can be rented for that particular peak time. Based on the number of requests there is always scope for extension, thus avoiding the problem of overloading.

Current developments in virtualization technology have transformed the way computing networks are designed and operated. Nonetheless, virtualization systems have a detrimental impact on the device output predictability, which poses many problems in managing efficiency and resource utilization. Lianjie Cao et al. [1] resolved efficiency issues by defining and modeling resource management task output in two technology scenario: distributed network simulation and network virtualization (NFV). Mahendra Bhatu Gawali et al. [2] proposed a heuristic approach combining the modified analytical hierarchy process (MAHP), bandwidth aware divisible scheduling (BATS) +BAR optimization, longest expected processing time preemption (LEPT) and dividing-and-conquering methods for scheduling tasks and allocating resources. Haitham Salman Chyad et al. [3] addressed utilization of soft computing with better efficiency on cloud to automate and plan the tools. Rajanikanth Aluvalu et al. [4] recommended productive allocation of virtual machines for successful use and increase in resource utilization and productive virtual machine implementation in cloud resources. Abhirup Khanna et al. [5] presented a novel resource allocation system which works on the dynamic resource allocation principles. B. Adrian et al. [6] proposed K-means clustering algorithm for virtual machine allocation process in place of cloudSim FIFO algorithm. Bhavan Bidarkar et al. [7] addressed optimal algorithms for load balancing. Sandeep Kapur et al. [8] presented a unique way of choosing encryption algorithms to prevent misuse. Mahesh et al. [9] suggested the method of decentralized distribution of services delegated to cloud consumers. The calculation of unequal usage of different VM services and the skew interest load balancing between VMs can be used.

Tinghuai Ma et al. [10] discussed five main cloud computing topics: locality-aware job scheduling; reliability-aware scheduling; RAS energy-aware; RAS layer of SaaS software; and workflow scheduling. Mayanka Katyal et al. [11] provided a broad variety of load balancing solutions in different cloud settings focused on Service Level Agreement (SLA) specifications. Lu Huang et al. [12] is focused on the actual condition in the cloud with the algorithms of resource

management and scheduling. V Vinothina et al. [13] discussed in detail various resource allocation strategies and their challenges.

HAProxy load balancer has very high load bearing capacity when implemented with Round Robin algorithm which uses basic data structures to perform resource allocation. The algorithm implemented in this work does not store the states of the previous requests of the prior states, instead it performs hash mapping of the resources allocated to a particular virtual machine. It also maintains the state list which shows the availability of virtual machine for performing resource allocation, based on which the HAProxy can allocate resources to the available virtual machines exclusively, thus maintaining data integrity. Load balancers like application load balancer redirects the incoming requests based on the IP address of the virtual machines whereas, the HAProxy load balancer is configured for the user to access the webpage through public DNS of the load balancer. The load balancer is responsible for allocating resources cyclically using round robin algorithm.

2.0 System Development

The system development starts by creating a Relational Database System (RDS). The RDS is linked to the two EC2 instances (VM1 and VM2), each one acting as a separate virtual server. HAProxy is set up as the load balancer as it is efficient with quick response time and it uses the RRA to assign resources to the VMs. Some more significant aspects of the virtual network include security groups which is created for each entity, and a VPC which is created for the whole network.

2.1 Methodology

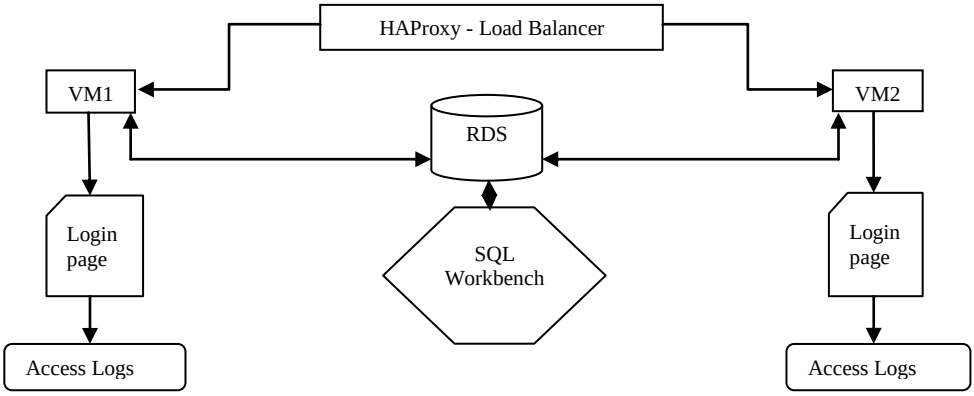

Fig. 1. Process of optimal load balancing using HAProxy

Process of optimal load balancing using HAProxy is shown in Fig. 1. Two Amazon Elastic Cloud Compute instances (VM1 and VM2) were created to act as virtual servers. Amazon RDS is used to create the database accessed by the EC2 instances. The contents of the RDS instance can be known by checking the SQL workbench. The VM1 and VM2 instances are connected to apache web

servers which host login pages that act as a user interface. A third instance EC2 instance is created to act as the HAProxy, which performs as a load balancer by distributing the load amongst the servers. The HAProxy instance is configured to balance the load by specifying the Public DNS of the virtual servers in its configuration file. Accessing the Public DNS of the HAProxy instance will redirect the user to either of the virtual servers. An entry is made into the access logs of each of the virtual servers whenever the HAProxy instance's Public DNS is visited. It is observed that even when one of the two virtual servers is turned off, the HAProxy redirects the user to the working server. The same will be reflected in the servers' access logs.

2.2 Creating Database Instance

Using the Amazon RDS on AWS, MySQL database is created. There is a header called Databases in AWS under which RDS is found. In order to set up MySQL database, certain specifications must be made while creating an instance of MySQL database. The class of instance used is t2.micro that has a storage capacity of 20 GB with one day of automated backup. The database is created by choosing Create database option. Various database engines are available on AWS including Oracle, Amazon Aurora, MySQL, etc. MySQL is used as the engine in this work for the database after configuration.

2.3 Creating EC2 Instances

Once the database is created using RDS instance, EC2 instances must be created to serve the purpose of web servers (VM) and load balancer. Two of the instances created host login pages via apache web server. Each instance created is linked to a previously created VPC and security group, having public and private subnets in at least two availability zones.

2.4 Installing Apache Web Server

After creating the EC2 instances they are connected to apache web servers that host login pages created as a PHP file. The option to fix all the previous bugs and to update security of the EC2 instance must be chosen to update the command prompt interface with a predefined command. In order to create the login page, it is necessary to install the software package of PHP using a command on Linux *sudo yum install.* Using this software package the Apache web server will be installed. If the apache web server page is displayed when the DNS of an instance is accessed, it can be confirmed that the instance is connected to the apache server. Permissions are given to the web server and www group is added to the ec2-user group. Other general commands for refreshing, logout, changing directory, etc. are specified.

2.5 Connecting Web Server and Database

Once the apache web server is installed it should be connected to the database created on AWS, i.e. the RDS instance. This connection is done by specifying parameters such as 'DB_SERVER' (the end point of the database), 'DB_USERNAME" (user dependent), 'DB_PASSWORD' (user dependent)

and 'DB_DATABASE' (default is sample) in the PHP file, which contains the code for the login page.

Connection with the created EC2 instance is kept as it is and a new subdirectory is created. The directory is changed to *www/html* and a fresh file is created in html format to code the login page in PHP. This file is edited in Sublime Text editor. The code ensures that a connection is established between the RDS instance and the Apache Web Server.

2.6 Round Robin Algorithm

RRA distributes workload on the basis on the round robin principle. The workload is distributed to the servers equally in a periodic manner. The assignment of the load is done in circular fashion without considering any priority and after reaching the last VM, it allocates the requests back to the first VM. The cycle repeats as long as the requests are being sent. This method of static allocation is easy to use and implement. The execution time for this model is less as there is no inter-process communication.

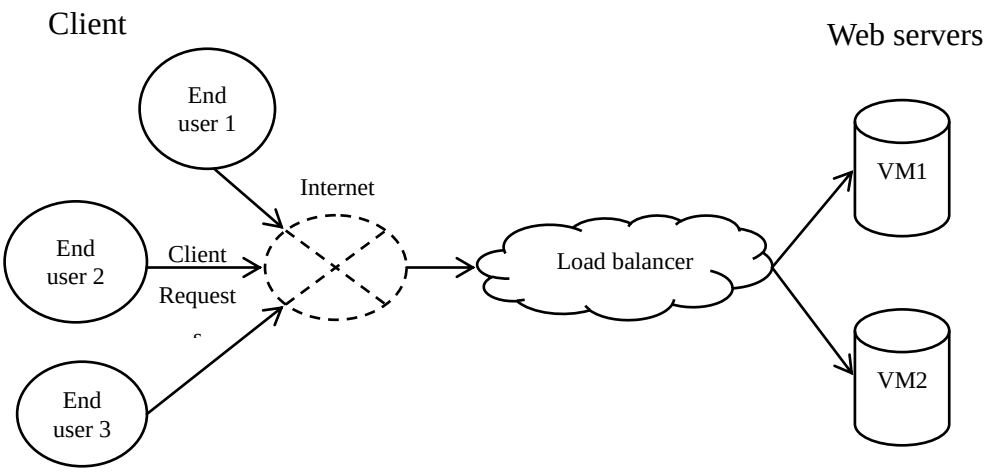

Fig. 3. Representation of interface from client to server in cloud environment

The RRA implemented in this work has a very high load bearing capacity. This algorithm need not save the prior states of the previous requests and is developed using concepts of basic data structures. One of the most important data structures maintained is the state list of each web servers (VM) which stores the allocation status of each VM, i.e., if it is busy or available for resource allocation. One of the other important data structures used is hash mapping, which stores the entry details of the request of a client and the VM allocated for the same [7]. If one of the VMs go down, the load balancer identifies it and the request is allocated to the VMs that are active and free to perform their duties. The first step of the algorithm is to consider all the VMs available in the list and set their allocation status to available or not available. At first there will be no entries and therefore the hash mapping will be set to no entries. If there are new requests received by the load balancer then the load

balancer must queue these requests first for allocation in RRA, and it removes requests one at a time from top of the queue of requests. If the VM satisfies the conditions of the request sent, and the hash map contains the data necessary, the VM is assigned for the mentioned process. If these conditions are not satisfied, the request is allocated to the following VM using the RRA. Once the allocation is done, an entry is made in the hash mapping data structure and also into the VM list. The process is shown in Fig. 3.

3.0 Results and Discussion

Public key is used to log into AWS via terminal, and the EC2 instance's Public DNS is used to access the login pages. Three EC2 instances were generated on AWS. Two instances (instance 1 and instance 2) act as virtual servers. The third instance is the HAProxy load balancer. To activate the HAProxy, it must be accessed using the security pem file, similar to that of the other two instances.

HAProxy is configured in the config file to balance the incoming load amongst the VMs. The Public DNS IDs of the two VMs (instance 1 and instance 2) are assigned as active servers to the HAProxy, and the RRA is specified. The login page is hosted by apache web server. All the login details are stored in the RDS instance created. The initial access log file is created as soon as the server is created, and it is located in the var/log/httpd folder.

Every time a user accesses the login pages, through the HAProxy, an entry is created in the access log file of that instance. Access log files are created on a daily basis, from which information on server load can be obtained. Simulation results are presented in Table 1.

Table 1. Simulation results based on the number of users

No of Users	Execution time, ms	Load average	Idle state	UNIX Ports	Internet connection state	Memory status
1-5	0.690	0.0~0.1	Almost 100	No duplicates	Established	Good
5-10	0.726	0.2	99			
10-15	0.747	0.7	92			
15-20	0.786	0.9	87			
20-25	0.801	1.2	81			

As the number of users increased, the average load on the load balancer, execution time and CPU usage increased. As the number of clients increased the run-time changed nominally, showing that the proposed model is responsive.

4.0 Conclusion

Resource allocation in a virtual environment was implemented using HAProxy as load balancer. Amazon RDS provides database storage of 20 GB which can be expanded to 1TB based on the requirements. The Round Robin algorithm used in this work is more efficient as the EC2 instances used as the web servers have same specifications of capacity and response time. HAProxy has high load bearing capacity with a low response time, ensuring faster resource allocation. The combination of HAProxy, EC2 instances and the Amazon RDS database makes the resource allocation mechanism efficient and reliable.

In case of web servers with different load bearing capacities, the round robin algorithm can be modified. Implementation of HAProxy as load balancer increases the flexibility and makes optimal usage of the VMs when compared with application load balancer.

References

1. L Cao, S Fahmy, P Sharma, Data-driven Resource Allocation in Virtualized Environments, *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA*, 20(6), 659-664, 2019

2. Mahendra Bhatu Gawali, Subhash K Shinde, Task scheduling and resource allocation in cloud computing using a heuristic approach, *Journal of Cloud Computing: Advances, Systems and Applications*, 7(4), 1-16, 2018

3. Haitham Salman Chyad, Raniah Ali Mustafa, Kawther Thabt Saleh, Study and Implementation of Resource Allocation Algorithms in Cloud Computing, *International Journal of Engineering & Technology*, 7 (4.28), 591 -594, 2018

4. Rajanikanth Aluvalu, M A Jabbar Vardhaman, Jalpa Kantaria, Performance evaluation of clustering algorithms for dynamic VM allocation in cloud computing, *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, 1560-1563*, 2017

5. A Khanna, Sarishma, RAS: A novel approach for dynamic resource allocation, *1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun*, 25-29, 2015

6. B Adrian, L Heryawan, Analysis of K-means algorithm for VM allocation in cloud computing, *International Conference on Data and Software Engineering (ICoDSE)*, Yogyakarta, 48-53, 2015

7. Bhavana Bidarkar, Shakheela Attikeri, Ramya S Pure, Round Robin Approach for Better VM Load Balancing in Cloud computing, *International Journal of Engineering Innovation & Research, 3 (*4), 499-502, 2014

8. Sandeep Kapur, Kumar Dinesh, Resource Utilization in Cloud Computing using Hybrid Algorithm, *Indian Journal of Science and Technology*, 9 (43), 1-10, 2016

9. Mahesh, Prashant, Poonam, An Efficient Dynamic Resource Allocation Strategy for VM Environment in Cloud, *International Conference on Pervasive Computing*, 4(2), 1-5, 2015

10. Tinghuai Ma, Ya Chu, Licheng Zhao, Otgonbayar Ankhbaya, Resource Allocation and Scheduling in Cloud Computing: Policy and Algorithm, *IETE Technical Review*, 31 (1), 4-16, 2014

11. Mayanka Katyal, Atul Mishra, A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment, *International Journal of Distributed and Cloud Computing*, 1 (2), 5-14, 2013

12. Lu Huang, Hai-shan Chen, Ting-ting Hu, Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing, *Journal of Software*, 8 (2), 480-486, 2013

13. V Vinothina, R Sridaran, Padmavathi Ganapathi, A Survey on Resource Allocation Strategies in Cloud Computing, *International Journal of Advanced Computer Science and Applications*, 3 (6), 97-104, 2012