International Journal of Engineering & Technology, 7 (4) (2018) 4607-4611

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi:14789
Research paper

A multi-node adaptive load balancing approach for availability in cloud computing

Hamza Kamal Idrissi [1] *, Ali Kartit [2]

[1] LRIT-CNRST, University Mohammed V, 4 Avenue Ibn Battouta. BP 1014 RP 10006 Rabat, Morocco
[2] LTI, University Chouaïb Doukkali –El Jadida, ENSAJ, Avenue Jabran Khalil Jabran BP 299 El jadida, Morocco
**Corresponding author E-mail: h.kamalidrissi@gmail.com*

Abstract

Cloud computing has become one of the most important fields in the IT technology domain. Its main objectives are to deliver different services for users, such as infrastructure, platform or software with a reasonable and more and more decreasing cost for the clients. In order to meet those objectives many aspects are studied and are subject to research, the availability is certainly one of the key sides of the Cloud Computing. Load balancing techniques deal with many aspects of the cloud such as performance, response time, … For the big distributed cloud systems that deal with many clients and big amounts of data and requests, load balancing is essential in order to properly satisfy all the demands. In this paper, we address the subject of availability with an adaptive load balancing approach in cloud computing. This proposed approach ensures load balancing as well as the availability of the system while avoiding points of failure.

Keywords: *Adaptive Load Balancing; Availability; Cloud Computing; Multiple Nodes.*

1. Introduction

Cloud computing is a new computing approach that strives for providing end users with computing services. It is based on some aspects of existing design principles, protocols, platforms ... Thus, it keeps some of the advantages of those technologies, and as a matter of fact, it comes with new capabilities that are useful for the programmability, scalability, and virtualization of resources.
Cloud Computing is confronted with the existing problems of its predecessor technologies, and comes with its own specific issues, for example regarding the availability and the load balancing aspects. Thus, depending on the load of the system and its nature, the rising of the final users' demands can be problematic and can cause the system to be slower or even unavailable. Due to the different characteristics of the requests and their flow, the load on each of the system's nodes can vary. As a result, all the nodes are not loaded with the same amount or need of demands, and the performance of the system is therefore reduced. This causes a significant dissimilarity in the response time to the requests as the ones assigned to overloaded nodes will be penalized while those assigned to lightly loaded nodes will be privileged in terms of the response time within the same cloud [1]. The system can even go to a state where it's unable to satisfy some of the demands which can be seen as a partial unavailability.
In order to overcome those issues of load imbalance and unavailability, a suitable load balancing technique is needed. Load balancing aims to avoid situations where some nodes are heavily loaded with work while other nodes are less or not at all working. It helps to satisfy the users and to use the resources efficiently [2]. In addition, if there is a node failure, the cloud system should be able to relocate the corresponding tasks which were meant to be processed be the failing node to another resource; this preserves the availability of the system without any impact on the end users.

In this paper, we will begin by describing the cloud computing concept. In the next section, we will describe some load balancing techniques, metrics and algorithms involved in the cloud environments. This description will not go deeply in the details of those techniques as the subject of this paper aims essentially to present a general approach. The third section details our proposed approach design and components. Last, we will identify some perspectives and some challenges that will remain to be addressed in the future.

2. Cloud computing

Computing is being converted to a model consisting of services that are commoditized and delivered in a manner similar to traditional utilities [3] such as water. As for Utility Computing is typically implemented using other computing infrastructure. In a cloud business model, a customer will pay the provider on a consumption basis, very much like the utility companies charge for basic utilities such as electricity, gas, and water, and the model relies on economies of scale in order to drive prices down for users and profits up for providers.
Cloud computing is therefore a new approach based on leveraging the Internet to consume software or other IT services on demand. End users share processing power, storage space, bandwidth, memory and software. With cloud computing, the resources are shared and so are the costs. Users can pay as they consume and only use what they need at any given time, keeping charges to the user cheap.

2.1. Cloud provider

The cloud model is composed of three types : [4]
Public clouds
This infrastructure can be used by the general public. This includes individuals, corporations and other types of organizations.

Typically, public clouds are ran by third parties or vendors over the Internet, and services are provided on pay-per-use basis. Public clouds are widely used in the development, deployment and management of enterprise applications, at reasonable costs. It delivers highly scalable and reliable applications rapidly but with a major, significant concern in public which is confidentiality.

Private clouds

The infrastructure is deployed within the frontiers of a same company and is used exclusively for the organization's profits. They are also called internal clouds and are mainly built by IT departments within enterprises who seek to enhance exploitation of infrastructure resources within the enterprise by provisioning the infrastructure with applications using the concepts of grid and virtualization. This preserves some aspects of cloud such as virtualization, availability of services and high levels of automation reducing the administrative overhead. However, the buying, maintenance and management of infrastructure is the responsibility of the company, which will increase operating costs. Community clouds refers to a specific subtype of public cloud in which several companies are sharing the same private cloud.

Hybrid clouds

A new concept combining resources from both internal and external providers will become the most popular choice for enterprises. For example, a company could select to use a public cloud service for general computing, but store its business critical data within its own data center. This may be because larger organizations are likely to have already invested heavily in the infrastructure required to provide in-house resources or they may be concerned about the security of public clouds.

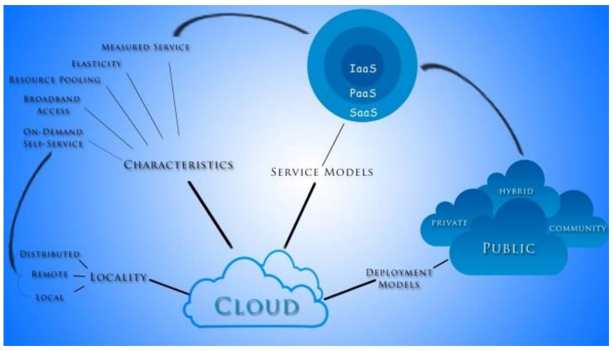

Fig. 1: View of the Cloud Computing Environment.

2.2. Cloud architecture

In many techniques, software such as Eucalyptus, OpenNebula and Nimbus are based on some common components. In a generic open-source cloud computing system, we can recognize six basic modules. First, we have the cloud control software whose aim is to bring together all cloud stack pieces and to ascertain enough abstraction so that a user can simply demand VMs with no harassment on how these components are created or coordinated.

Secondly, hardware, network and operating systems that are on the various physical machines in the system. It should be virtualized or paravirtualized depending of the virtualization framework compatibility. Paravirtualization is not adopted unless the framework could not handle the physical machines. The network includes the DNS, DHCP and the subnet organization of the physical machines. It also embraces virtual bridging and networking of the network that is required to give each VM a unique virtual MAC address. This bridging is done with the help of programs like bridge-utils, iptables or ebtables.

The third module is the virtual machine hypervisor, AKA Virtual Machine Monitor (VMM). Popular VMMs consist of Xen, KVM and VirtualBox, which are open-source, and VMware which is commercial. These programs afford a software which allows VMs to run. In order to start and stop a VM, VMMS relies on a library called Libvirt.

The fourth component is a repository of disk images that can be copied and used as the basis for new virtual disks. In any specified cloud, we must make a difference between template disk images and runtime disk images. Also, when a VM is laid, one of those templates copied and is wrapped into a disk image appropriate for the given hypervisor. Usually, this includes adding a swap partition and resizing the disk image to the appropriate size.

The fifth component is the front-end for users. Represented by an interface for users to request virtual machines, specify their parameters, and obtain needed credentials and keys to sign in.

The last module is the cloud framework itself, where Eucalyptus, OpenNebula and Nimbus are located. This framework analyses inputs from the front-end, recovers the needed disk images from the archive, signals a VMM to run a VM and then mention to DHCP and IP bridging programs to assign MAC and IP addresses for the VM [5].

3. Load balancing and objectives

3.1. Load balancing definition and goals

Load balancing represents the fact of equally distributing the load among several resources in a distributed or parallel system in order to equalize workloads effectively and to enhance the execution time of a task [6], [7]. In fact, it avoids a situation where some of the nodes are heavily loaded while other nodes are not doing any work. Load balancing guarantees that all the processors in the system or every node in the network process approximately an equal amount of work at any instant of time.

This concept involves first decomposing the overall computation into tasks and then assigning the tasks to nodes [8]. The decomposition and assignment steps together are often called partitioning. The optimization objective for partitioning is to balance the load among nodes and to minimize the internodes communication needs. Executing a task in this distributed environment requires mapping the processes to nodes. The number of resources generated by the partitioning step may not be equal to the total number of nodes. Thus a node can be idle or loaded with multiple jobs.

As demonstrated before and based on [9], [10], load balancing aims are:

To enhance the performance significantly

To redistribute the node tasks in case the latter suffers from overloading, malfunction or failure

To uphold the system stability

Scalability and flexibility: the distributed system in which the algorithm is implemented may change in size or topology. So the algorithm must be scalable and flexible enough to allow such changes to be handled easily.

Priority: prioritization of the resources or jobs needs to be done on beforehand through the algorithm itself for better service to the important or high prioritized jobs in spite of equal service provision for all the jobs regardless of their origin.

3.2. Load balancing and cloud computing

The fundamental clue about cloud computing is to afford assets such as VMs as services on demand. Assigning effective VM on demand is being carried out with the support of the load balancing algorithms in the cloud computing [10]. As the load balancing algorithm plays an important role while determining which VM is to be allocated on demand to the user.

Cloud vendors then are based on automatic load balancing services [11], which allow clients to increase the number of CPUs, memory or hard disk for their resources in order to scale with bigger demands. This service is implied and depends on the clients' professional requirements. So, load balancing supplies two important basics, mostly to promote availability of Cloud resources and secondarily to uphold a global performance, energy is saved in case of under loading.

A perfect load balancing designed for cloud service should circumvent overloading or under loading of any specific node. So the selection of load balancing algorithm is not unproblematic because it involves supplementary restraints like security, trustworthiness, throughput, etc. Consequently, the main aim of a load balancing algorithm in a cloud computing environment is to improve the response time of job by simplifying interaction between the nodes, choosing nature of work to be transferred and selecting the possible nodes which could hold the task or the process to be moved in case of failure of its hosting node.

4. Algorithms and challenges

4.1. Load balancing metrics

The goal of load balancing is to effectively distribute the work load between available resources, in order to maximize the benefit from those resources and to have quick computing and processing for the client requests. It is done so as to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded [12].

To evaluate the quality of a load balancing technique, architecture or system, many metrics can be used. Some of them must be maximized while others should be minimized, in order to have an efficient load balancing system. The most common are described by Dash, M et al in [13], as follows:

Throughput is used to calculate the no. of tasks whose execution has been completed. It should be high to improve the performance of the system.

Overhead Associated determines the amount of overhead involved while implementing a load-balancing algorithm. It is composed of overhead due to movement of tasks, inter-processor and inter-process communication. This should be minimized so that a load balancing technique can work efficiently.

Fault Tolerance is the ability of an algorithm to perform uniform load balancing in spite of arbitrary node or link failure. The load balancing should be a good fault-tolerant technique.

Migration time is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system.

Response Time is the amount of time taken to respond by a particular load balancing algorithm in a distributed system. This parameter should be minimized.

Resource Utilization is used to check the utilization of resources. It should be optimized for an efficient load balancing.

Performance is used to check the efficiency of the system. This has to be improved at a reasonable cost, e.g., reduce task response time while keeping acceptable delays.

4.2. Load balancing algorithms

In order to achieve a fair resource allocation between demanding tasks and to have a certain performance in the overall clod system, load balancers resort to various scheduling algorithms. Since research in this field is still ongoing, no technique can be considered as the best. Hence, the chosen algorithm to utilize depends on many considerations; especially the size of the cloud system, the nature of the requests, the amount of available resources … Those algorithms can be classified in various ways. Thus, they can be classified as static or dynamic. A static load balancing algorithm does not take into account the previous state or behavior of a node while distributing the load. On the other hand, a dynamic load balancing algorithm checks the previous state of a node while distributing the load. The dynamic load balancing algorithm is applied either as a distributed or non-distributed [14]. Load balancers can work in two ways: one is cooperative and non-cooperative. In cooperative, the nodes work simultaneously in order to achieve the common goal of optimizing the overall response time. In non-cooperative mode, the tasks run independently in order to improve the response time of local tasks [14].

The most known algorithms are listed by Mathew, T et al [15] in Table 1. As shown in this table, each algorithm has its own characteristics, advantages and disadvantages. Thus, it is depending on the context that some algorithm or another is chosen to perform load balancing, such as the type of requests, whether they are all of the same type or not,...

Table 1: Load Balancing Algorithms Characteristics

Scheduling Method	Parameters Considered	Advantages	Disadvantages
First Come First Serve	Arrival time	Simple in implementation	Doesn't consider any other criteria for scheduling
Round Robin	Arrival time, Time quantum	Less complexity and load is balanced more fairly	Preemption is required
Opportunistic Load Balancing	Load balancing	Better resource Utilization	Poor makespan
Minimum Execution Time Algorithm	Expected execution time	Selects the fastest machine for scheduling	Load Imbalanced
Minimum Completion Time Algorithm	Expected completion time, Load balancing	Load balancing is Considered	Optimization in selection of best resource is not there
Min-Min, Max-Min	Makespan, Expected completion time	Better makespan compared to other algorithms	Poor load balancing and QoS factors are not considered
Genetic Algorithm	Makespan, Efficiency, Performance, Optimization	Better performance and efficiency in terms of makespan	Complexity and long time consumption
Simulated Annealing	Makespan, Optimization	Finds more poorer solutions in large solution space, better makespan	QoS factors and heterogeneous environments can be considered
Switching Algorithm	Makespan, Load balancing, Performance	Schedules as per load of the system, better makespan	Cost and time consumption in switching as per load
K-percent Best	Makespan, Performance	Selects the best machine for scheduling	Resource is selected based on the completion time only
Suffrage Heuristic	Minimum completion time, Reliability	Better makespan along with load balancing	Scheduling done is only based on a suffrage value
Benefit Driven, Power Best Fit, Load Balancing	Energy Consumption, Cost, Load balancing	Power consumption is reduced and cost is reduced even more number of servers used	Other QoS factors and completion time of tasks are less considered
Energy efficient method using DVFS	Energy Consumption, Makespan, Execution time	Energy saving as per load in the system producing better makespan	Cost and implementation complexity can make better in future
DENS	Traffic load balancing, Congestion, Energy Consumption	Communication load is considered and job consolidation is done to save energy	Consider only data intensive applications with less computational needs
e-STAB	Energy efficiency, Network awareness, QoS, performance	Load balancing and energy efficiency is achieved based on traffic load, congestion and delay are avoided	QoS factors can be considered for improvement in overall performance
Task Scheduling & Server Provisioning	Energy Consumption, Task response time, Deadline	Energy is reduced meeting the deadline of tasks	Makespan and cost are less considered here
Improved Cost Based Algorithm	Processing cost, Makespan	Resource cost and computation performance is considered before scheduling	Dynamic cloud environment and other QoS attributes are not considered
Priority based Job Scheduling Algo-	Priority of tasks, Expected com-	Priority is considered for schedul-	Makespan, consistency and com-

rithm	pletion time	ing. Designed based on multiple criteria decision making model	plexity of the proposed method can be considered for improvement
Job Scheduling based on Horizontal Load Balancing	Fault tolerance, Load balancing, Response time, Resource utilization, Cost, Execution time	Probabilistic assignment based on cost. Highest probable resource and task are selected for assignment.	Algorithm never mentions how the total completion time of the tasks will remain
User Priority guided Min- Min	Priority , Makespan, Resource Utilization, Load balancing	Prioritized is given to users improving load balancing and without increasing total completion time.	Rescheduling of tasks to perform load balancing will increase the complexity and time
WLC based Scheduling	Load balancing, Efficiency, Processing Speed	Dynamic task assignment strategy proposed, task heterogeneity is considered	Considering only load balancing feature
Cost Based Multi QoS Based DLT scheduling	Load balancing, Makespan, QoS, Performance, Cost	DLT based optimization model is designed for getting better overall performance	Machine failure, communication overheads and dynamic workloads are not considered
Enhanced Max-Min Algorithm	Makespan, Load balance, Average execution time	Improves makespan and load balancing when large difference occurs in the length of longest task and other tasks or speed of processors	Parameters considered are limited and only theoretical analysis is performed

The aforementioned metrics and algorithms have the goal of meeting the challenges that load balancing in cloud computing has to deal with. Thus, to achieve that, the following objectives must be reached:

Improving the overall performance, which is the main objective of the load balancing as a whole.

Avoiding starvation for the processed tasks, this can be done by improving the throughput and minimizing the response as well as the migration times.

Reliability, which can be obtained by a fault tolerant approach.

Security matters, which is an important side of the system since data flows and communication have to be protected from any kind of undesirable activities.

To realize those objectives while taking into consideration the associated challenges, we proposed a multi cluster approach. The next section details this approach and describes how it deals with the stated matters.

5. Proposed approach

With the spreading of usage of cloud computing and the growing demand of delivered services via this new technology, numerous techniques and approaches are proposed by researchers to deal with this demand. Most works tend to work on load balancing algorithms to enhance performance and allocate available resources as efficiently as possible. Still, due to the heterogeneity and disparities between environments and architectures, all techniques leave some problems unsolved. Our objective is to resolve some of those left issues with our proposed approach. The issues we aim at dealing with are: efficient load balancing between the different resource nodes that process the client tasks, in a secure way as well as the elimination of possible single point of failure in a semi centralized load balancing architecture. Another addressed matter is the continuous use of all available resources, in fact, no resource should remain unused due to any kind of problem, such as the failure of the central load balancer for example, that can lead to the partial or the total shut down of the system (resource nodes). Thus, the proposed method has the advantage of efficient load balancing, continuous availability if any failure is experienced in a central load balancer and secured data flows between the system nodes. The detailed description of the aforementioned architecture is as follows:

The load balancing elements are: the main load balancer (MLB, it's the central node), the resources' clusters and an authentication element, in order to have secure data flows between the two previous elements.

The main load balancer can be seen as the system node coordinator, it receives task processing requests from the clients, and it applies an algorithm to match each client request to a suitable cluster to process it. It has its own table of clusters, each cluster with a weight representing its average processing capacity, thus the main load balancer can choose the right cluster to transfer a client request to and the load is fairly shared between all the clusters. No job queue is needed at this level as each cluster has its own local queue.

Each resources' cluster consists of a group of nodes (resources), it has its own local load balancer (LLB), and this latter receives requests from the MLB and deals with the load balancing inside its local cluster. It has a queue where the upcoming jobs are stored until a node inside the cluster is available to process them. The local load balancers use a scheduling algorithm to perform the load balancing, they also keep a resource table that is constantly updated if there is any change in the corresponding cluster nodes (for example, in case of the failure of a node, the addition of a new node,...). Depending on the changes in this table, the weight of the cluster is updated in the MLB table. The local load balancer table also contains the weight of each node, hence the local load is fairly balanced between the cluster elements.

To ensure the availability of the system and to eliminate single points of failure, another proposed functionality of this architecture is to make each LLB an MLB candidate, it will replace the MLB in the case it comes to fail. In that case, the first LLB that is aware of the MLB failure, becomes the new MLB, it notifies the other LLBs of this change, gathers data about the other LLBs, dispatches its clusters' nodes between the other clusters and starts its work as the new MLB by sending the elements in its queue to the other LLBs. This ensures the continuous availability of the system while the resources are efficiently used.

We also manage to add an authentication layer of security, so that only authenticated users could send jobs through our load balancing architecture. With this way, no overloading should occur within the platform and the load balancing architecture should stick to the cloud principles such as paying the minimum possible so we stop wasting resources. The authentication server should also grant special priority to a user and then his jobs shall be executed as quickly as possible.

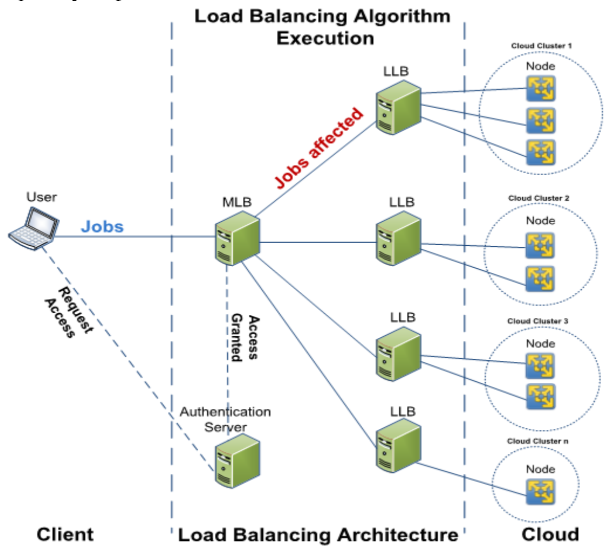

Fig. 2: Adaptive Load Balancing Approach.

6. Conclusion

The recent enthusiasm for the use of Cloud Computing related services has brought to light many important issues that the providers have to deal with; one of the most important is certainly availability and performance. To face those problems, some techniques have been proposed and this area is still subject to research. Load balancing is a key method to ensure a good overall performance in Cloud systems and provide efficient resource usage. In fact, Load Balancing can intervene in many aspects of Cloud Computing, such as availability for example. In this paper, we addressed some characteristics of Load balancing in cloud computing, namely its critical issues and challenges, especially the availability side. We also presented an adaptive approach to benefit from load balancing in order to overcome availability issues in a Cloud Computing environment using an adaptive architecture.

The proposed method deals with availability as well as with the system performance while taking into account fault tolerance elements. In future work, we intend to explore this method possibilities in different cloud computing environments in terms of resources and demands. Other possible enhancements can be studied as well as some other aspects such as cost effectiveness, energy consumption,

References

[1] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," presented at the Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on, 2012, pp. 3–8.

[2] N. J. Kansal and I. Chana, "Cloud load balancing techniques: A step towards green computing," *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 1, pp. 238–246, 2012.

[3] H. K. Idrissi, A. Kartit, and M. E. Marraki, "FOREMOST SECURITY APPREHENSIONS IN CLOUD COMPUTING," *J. Theor. Appl. Inf. Technol.*, vol. 59, no. 3, pp. 580–588, Jan. 2014.

[4] H. Kamal Idrissi, A. Kartit, and M. El Marraki, "A taxonomy and survey of Cloud computing," presented at the Security Days (JNS3), 2013 National, 2013, pp. 1–5. https://doi.org/10.1109/JNS3.2013.6595470.

[5] P. Sempolinski and D. Thain, "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus," presented at the Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010, pp. 417–426. https://doi.org/10.1109/CloudCom.2010.42.

[6] A. S. N and M. Hemalatha, "An Approach on Semi-Distributed Load Balancing Algorithm for Cloud Computing System," *Int. J. Comput. Appl.*, vol. 56, no. 12, pp. 5–10, Oct. 2012.

[7] R. P. Padhy, "Load balancing in cloud computing systems," National Institute of Technology, Rourkela, 2011.

[8] C. Xu and F. C. Lau, *Load Balancing in Parallel Computers: Theory and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[9] A. M. Alakeel, "A guide to dynamic load balancing in distributed computer systems," *Int. J. Comput. Sci. Inf. Secur.*, vol. 10, no. 6, pp. 153–160, 2010.

[10] M. M. D. Shah, M. A. A. Kariyani, and M. D. L. Agrawal, "Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm," *Int. J. Comput. Sci. Inf. Technol. Secur. IJCSITS ISSN*, pp. 2249–9555, 2013.

[11] Soumya Ray, "Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment," *Int. J. Cloud Comput. Serv. Archit.*, vol. 2, no. 5, pp. 1–13, Oct. 2012.

[12] R. Gupta and R. Bhatia, "An Enhanced and Secure Approach of Load Balancing in Cloud Computing," 2014.

[13] M. Dash, A. Mahapatra, and N. R. Chakraborty, "Cost Effective Selection of Data Center in Cloud Environment," *Int. J. Adv. Comput. Theory Eng. IJACTE*, vol. 2, pp. 2319–2526, 2013.

[14] S. S. Moharana, R. D. Ramesh, and D. Powar, "Analysis of load balancers in cloud computing," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 101–108, 2013.

[15] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, 2014, pp. 658–664. https://doi.org/10.1109/ICACCI.2014.6968517.