DYNAMIC LOAD BALANCING IN CLOUD STORAGE

Project Report submitted in fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science & Engineering

By

Shagun Dogra (121283)

Under the supervision of

Mr. Punit Gupta

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Dynamic Load Balancing in Cloud Storage"** in fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2015 to May 2016 under the supervision of **Mr. Punit Gupta** (Assistant Professor (Grade-I), Information Technology). The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Student Name: Shagun Dogra
Student Roll No:121283

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Supervisor Name: Mr. Punit Gupta
Designation : Assistant Professor (Grade-I)
Department name : Information Technology
Dated :

Acknowledgement

I would like to use this opportunity to express my gratitude to everyone who supported me throughout the course of this B.Tech project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and inspiring views on a number of issues related to the project.

I am especially grateful to **Mr. Punit Gupta**, Project Supervisor, for his valuable suggestions, support and constant encouragement during the course of the project. His perpetual energy, motivation, enthusiasm and immense knowledge inspired me to discipline myself in efficiently executing my multiple responsibilities simultaneously.

Date:

Shagun Dogra (121283)

Table of Contents

List of Figures

List of Tables

List of Graphs

List of Abbreviations

QoS	Quality of service
VM	Virtual machine
PaaS	Platform as a Service
SaaS	Software as a Service
IaaS	Infrastructure as a Service
HaaS	Hardware as a Service
API	Application Program Interface
FCFS	First Come First Serve
RR	Round Robin
DVFS	Dynamic Voltage and Frequency Scaling
IQR	Inter Quartile Range
AWS	Amazon Web Services
LBMM	Load Balancing Min-Min
SLA	Service Level Agreement
ABC	Artificial Bee Colony
LBACO	Load Balancing Ant Colony
ACO	Ant colony
LBA	Load Balancing Algorithms
UAA	utilization after allocation

Abstract

"Cloud computing" is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc.

From few last decades, there is a rapid growth of data in cyberspace therefore cloud is becoming an important service in internet computing. Infrastructure as a Service provides on-demand virtual machines to users. In accordance with the client's demand at a datacenter, host creates VMs which increase the load on the host. If the load of any host exceeds its capacity, then it affects their efficiency. Load balancing plays an important role in the deployment of virtual machines onto physical hosts. Resource requirement of virtual machine is hard to predict. Thus to deal with this issue, lots of load balancing methods are present. So in this paper we discuss these load balancing methods to provide overview of latest approaches in IaaS. Different load balancing methods have different parameters. So, we compared them on the basis of parameters used for their method. We have proposed our own load balancing method after comparing all available methods.

CHAPTER 1
INTRODUCTION

Cloud computing is emerging as a new paradigm of large scale distributed computing. It has moved computing and data away from desktop and portable PCs, into large data centres [1]. It is one of the technologies that provide cloud storage to manage the data. Cloud storage act as a repository in which the data is maintained, managed and is made available to the end users. Large generated application datasets can flexibly be stored or deleted in the cloud and from here end users' access this data by using cloud storage services interface, without accessing any storage server in real. It has widely been adopted by the industry, though there are many existing issues like Load Balancing, Virtual Machine Migration, Server Consolidation, Energy Management, etc. that are not fully addressed to which Central to these issues is the issue of load balancing that is a mechanism to achieve a high user satisfaction and resource utilization ratio by distributing the dynamic workload evenly across all the nodes in the whole cloud.

In this work, we have proposed the load balancing approach to balance the load in terms of requests of overloaded servers in the cloud storage.

1.1. Cloud Computing

Cloud computing is computing in which large groups of remote servers are networked to allow centralized data storage and online access to computer services or resources. Cloud computing, or in simpler shorthand just "the cloud", also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users.

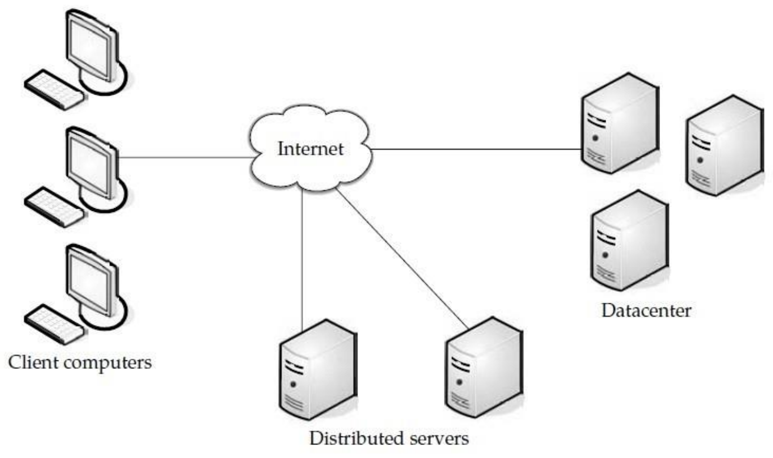


Figure 1: Three components make up a cloud computing solution [2]

1.1.1. Service Models

Service means different types of applications provided by different servers across the cloud. It is generally given as "as a service". Cloud service delivery is divided into three models. The three service models are [2]:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

1.1.1.1 Software as a Service (SaaS)

In SaaS, the user uses different software applications from different servers through the Internet. The user uses the software as it is without any change and it doesn't require integration to other systems. The provider does all the upgrades and patching while keeping the infrastructure running [2].

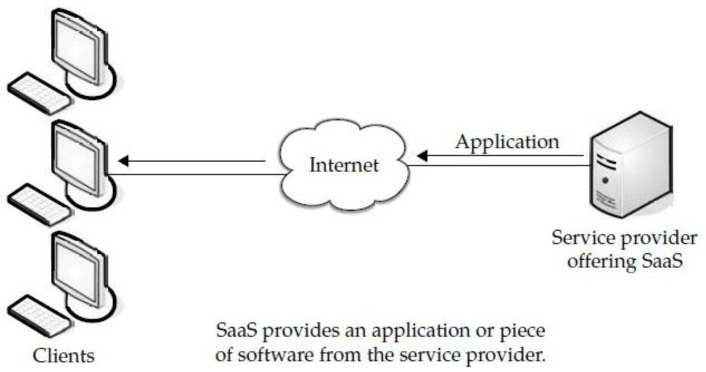

Application
Internet
Service provider offering SaaS
Clients
SaaS provides an application or piece of software from the service provider.

Figure 2: Software as a service (SaaS)[2]

1.1.1.2 Platform as a Service (PaaS)

PaaS provides all the resources that are required for building applications and services completely from the Internet, without downloading or installing software [2]. PaaS services are software design, development, testing, deployment, and hosting. Other services can be team collaboration, database integration, web service integration, data security, storage and versioning etc.

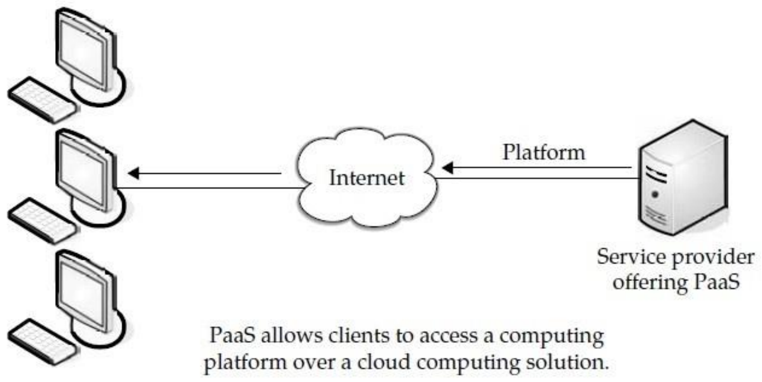

Platform
Internet
Service provider offering PaaS
PaaS allows clients to access a computing platform over a cloud computing solution.

Figure 3: Platform as a service (PaaS) [2]

1.1.1.3 Infrastructure as a Service (IaaS)

IaaS is also known as Hardware as a Service (HaaS). In this infrastructure or actual hardware is provided to customers who are responsible to install operating systems and necessary software as per their usage. IaaS cloud is usually provided to users in the form of Virtual Machines (VMs), such as Amazon EC2. In an IaaS cloud, users can apply VMs on-demand to deploy and run their applications also provide services to their clients which will be helpful to clients.[3]

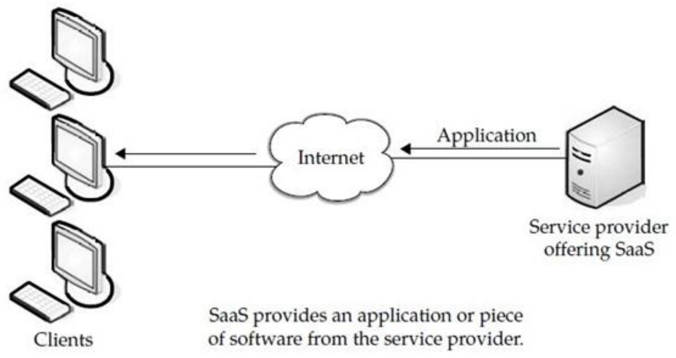

Application
Internet
Service provider offering SaaS
Clients
SaaS provides an application or piece of software from the service provider.

Figure 4: Hardware as a service (HaaS) [2]

1.1.2.Deployment Mode

Cloud computing model can be categories into three deployment models:

- Public Cloud
- Private Cloud
- Hybrid Cloud

1.1.2.1 Public Cloud

A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free or offered on a pay-per-usage model. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure at their data center and access is generally via the Internet.

1.1.2.2 Private Cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally. Undertaking a private cloud project requires a significant level and degree of engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources.

1.1.2.3 Hybrid Cloud

Hybrid cloud is a composition of two or more clouds (private, community or public) that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources.

1.1.3. Layered Architecture of Cloud Computing

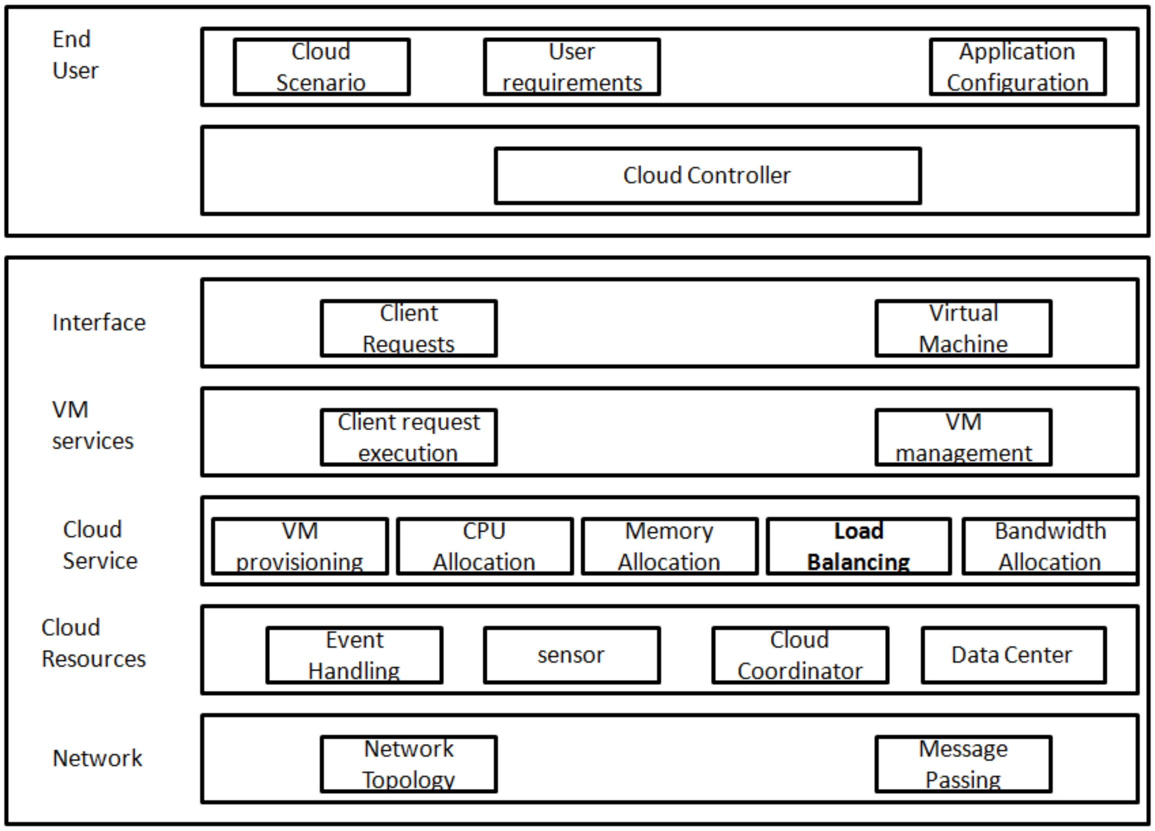


Figure 5: Layered Architecture of Cloud Computing[3]

Here figure 1.1 shows the layered architecture of cloud computing. In figure 1.1, load balancing is one of the cloud services in cloud computing.

1.1.4. Characteristics

Cloud computing exhibits the following key characteristics:

- ***Agility*** improves with users' ability to re-provision technological infrastructure resources.
- **API** accessibility to software that enables machines to interact with cloud software in the same way that a traditional user interface (e.g., a computer desktop) facilitates interaction between humans and computers. Cloud computing systems typically use (REST)-based APIs.
- ***Cost reductions*** claimed by cloud providers. A public-cloud delivery model converts capital expenditure to operational expenditure which lowers barriers to entry, as infrastructure is typically provided by a third party and does not need to be purchased for one-time or infrequent intensive computing tasks.
- ***Device and location independence*** enable users to access systems using a web browser regardless of their location or what device they use (e.g., PC, mobile phone). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect from anywhere.
- ***Maintenance*** of cloud computing applications is easier, because they do not need to be installed on each user's computer and can be accessed from different places.
- ***Productivity*** may be increased when multiple users can work on the same data simultaneously, rather than waiting for it to be saved and emailed. Time may be saved as information does not need to be re-entered when fields are matched, nor do users need to install application software upgrades to their computer.
- ***Reliability*** improves with the use of multiple redundant sites, which makes well-designed cloud computing suitable for business continuity and disaster recovery.
- ***Scalability and elasticity*** via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis in near real-time, without users having to engineer for peak loads.

The National Institute of Standards and Technology's definition of cloud computing identifies "five essential characteristics": [4]

1) ***On-demand self-service:*** A business will secure cloud-hosting services through a cloud host provider which could be your usual software vendor. You have access to your services and the power to change cloud services through an online control panel or directly with the provider. You can add or delete users and change storage networks and software as needed.
2) ***Broad network access***.Your team can access business management solutions using their smartphones, tablets, laptops, and office computers. They can use these devices wherever they are located with a simple online access point.
3) ***Resource pooling***. The cloud enables your employees to enter and use data within the business management software hosted in the cloud at the same time, from any location, and at any time. This is an attractive feature for multiple business offices and field service or sales teams that are usually outside the office.
4) ***Rapid elasticity***. If anything, the cloud is flexible and scalable to suit your immediate business needs. You can quickly and easily add or remove users, software features, and other resources.
5) ***Measured service***. Going back to the affordable nature of the cloud, you only pay for what you use. You and your cloud provider can measure storage levels, processing, bandwidth, and the number of user accounts and you are billed appropriately. The amount of resources that you may use can be monitored and controlled from both your side and your cloud provider's side which provides transparency.

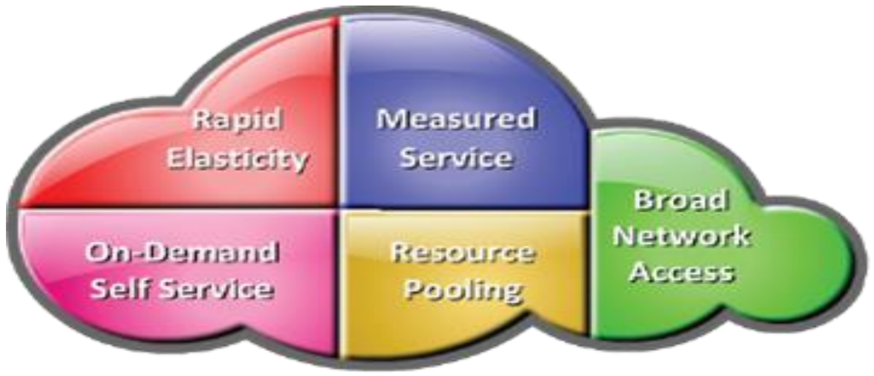


Figure 6: Characteristics of Cloud[3]

1.1.5. Virtualization

It is a very useful concept in context of cloud systems. Virtualisation means “something which isn’t real”, but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine. Virtualisation is related to cloud, because using virtualisation an end user can use different services of a cloud. The remote data center will provide different services in a fully or partial virtualised manner. The two types of virtualization are found in case of clouds as given in [2] :

1) Full Virtualization
2) Para Virtualisation

1.1.5.1 Full Virtualisation

In case of full virtualisation a complete installation of one machine is done on the another machine. It will result in a virtual machine which will have all the software’s that are present in the actual server. Here the remote datacenter delivers the services in a fully virtualised manner.

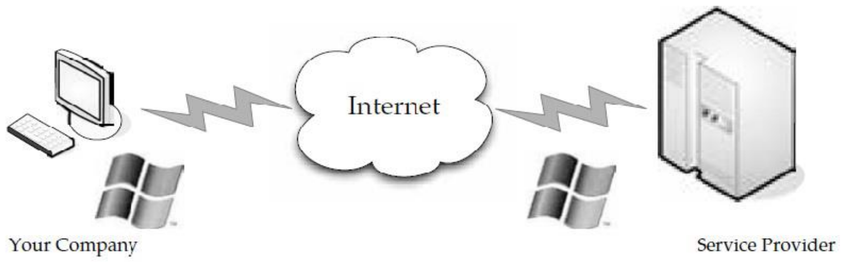


Figure 7: Full Virtualization[2]

Full virtualization has been successful for several purposes as pointed out in [2]:

- Sharing a computer system among multiple users.
- Isolating users from each other and from the control program
- Emulating hardware on another machine

1.1.5.2 Para virtualization

In par virtualisation, the hardware allows multiple operating systems to run on single machine by efficient use of system resources such as memory and processor. E.g. VMware software. Here all the services are not fully available, rather the services are provided partially.

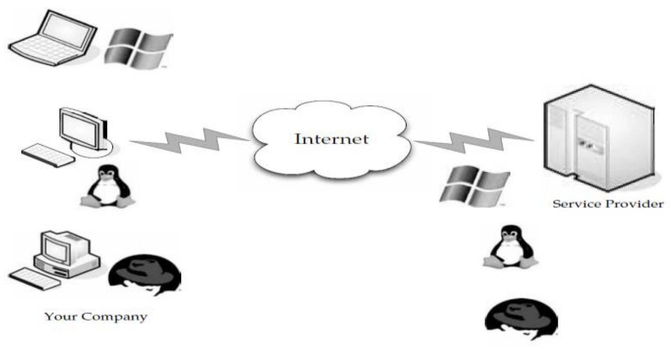


Figure 8: Para Virtualization [2].

Para virtualization has the following advantages as given in [2]:

- **Disaster recovery:** In the event of a system failure, guest instances are moved to another hardware until the machine is repaired or replaced.
- **Migration:** As the hardware can be replaced easily, hence migrating or moving the different parts of a new machine is faster and easier.
- **Capacity management:** In a virtualised environment, it is easier and faster to add more hard drive capacity and processing power. As the system parts or hardware can be moved or replaced or repaired easily, capacity management is simple and easy.

1.2 Cloud Storage

Storage of data over the Internet is one of the essential application of cloud computing. Cloud storage is capable of providing the users to store the enterprise data in the different storage servers of different vendors instead of storing the data in particular storage server. Cloud storage implements the location transparency, so that user can never know where his data stored in the cloud storage but it provides the abstract view of local storage. Cloud storage is simply an alias used to pointing out to virtual storage in the cloud environment.

Hence in cloud storage, client’s data can be accumulated on one or many of the systems that participate in the cloud environment. But the real repository area may significantly vary from time to time or even moment to moment, as the cloud powerfully oversees accessible storage areas. Anyway, despite the fact that the storage place is virtual, the client gets a static view of his data area and can trivially work with his cloud storage which physically resides far away from the client [5].

Economically, in comparison to dedicated physical resource, virtual resources are less expensive. Concerning security, enterprise data put away in the cloud storage is safe from unintentional eradication or equipment failures, on the grounds that it is replicated over numerous physical devices. Because various replicas of the information are kept ceaselessly, the cloud storage keeps on working as typical regardless of the possibility that one or more machines get disconnected from the net [5].

1.3 Load Balancing

It is a mechanism that distributes the dynamic local workload evenly across all the nodes in the whole cloud to avoid a situation where some nodes are heavily loaded while others are idle or doing little work. It helps to achieve a high user satisfaction and resource utilization ratio, hence improving the overall performance and resource utility of the system. It also ensures that every computing resource is distributed efficiently and fairly[4]. It further prevents bottlenecks of the system which may occur due to load imbalance. When one or more components of any service fail, load balancing helps in continuation of the service by implementing fair-over, i.e. in provisioning and de-provisioning of instances of applications without fail. The goal of load balancing is improving the performance by balancing the load among these various resources (network links, central processing units, disk drives.) to achieve optimal resource utilization, maximum throughput, maximum response time, and avoiding overload. To distribute load on different systems, different load balancing algorithms are used.

1.3.1. Goals of Load balancing

In order to balance the requests of the resources it is important to recognize a few major goals of load balancing algorithms:

a) *Cost effectiveness*: primary aim is to achieve an overall improvement in system performance at a reasonable cost.

b) *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.

c) *Priority*: prioritization of the resources or jobs need 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.

1.3.2. Metrics for Load Balancing

Various important matrices used for load balancing algorithms are discussed as follow:[6]

- ***Throughput*** is utilized to ascertain the number of job whose processing has been accomplished. It ought to be maximized to enhance the execution of the system.
- ***Overhead Associated*** decides the measure of cost of actualizing a load balancing technique in term of time. It is made out of extra cost because of migration of jobs, between various process interactions and processors. This ought to be least so a load adjusting strategy can perform effectively.
- ***Fault Tolerance*****:** The capacity of an algorithm to perform uniform load balancing despite subjective nod or connection failure. The load balancer ought to be a fault tolerant strategy.
- ***Migration time*****:** The total duration to move the tasks or processes among the nodes available in the system. It ought to be least in order to upgrade the execution of the system.

- ***Response Time*:** The measure of time elapsed to react by a specific load adjusting mechanism in a disseminated system. It ought to be least.
- ***Resource Utilization*:** It is utilized to analyses the use of resources. It ought to be advanced for an effective burden adjusting.
- ***Scalability:*** It is the capacity of an algorithm to operate load balancing efficiently the size of system increased or decreased. It ought to be enhanced.
- ***Performance:*** It is utilized to check the effectiveness of the system. This must be enhanced at a sensible expense, e.g. minimize tasks response time while keeping worthy deferrals.

1.4. Problem Statement

As we can see that from past few decades, there is a huge proliferation of data in cyberspace. In order to manage data efficiently, cloud provides remote services to subscribed clients through Internet. . In this system few of the storage server gets huge clients requests where as other servers remain idle or least loaded. This unequal distribution of load on servers leads to degradation of the performance of overall system and increases the response time of submitted requests. If the load of any server exceeds its capacity, then it affects their efficiency. Load balancing plays an important role in the deployment of virtual machines onto physical hosts. Handling this challenge related to the load balancing in the cloud keeping in account the energy efficiency and scalability of the system and develop a method which minimizes the response time and maximizes the overall throughput of the system is our main concern.

1.5. Objective

- Designing a scalable and energy-efficient load balancing algorithm for handling dynamic load on the hosts.
- Enhances the power efficiency of the system as well as takes into consideration the system scalability.

- Reduces the waiting time of client requests in server queue for processing.
- Enhance the utilization of server.
- Reduce the overall response time of system.

1.6. Organization of report

This report is organized into five chapters.

Chapter 1 describes what is cloud computing, various core services provided by cloud computing, its service and deployment models, cloud storage, load balancing in cloud storage, goals and metrics of load balancing, problem statement and objective of report.

Chapter 2 describes about the previous research work related to the proposed problem statement.

Chapter 3 describes about the proposed work, simulation of existing algorithms of countering load balancing and studying their outcomes.

Chapter 4 describes about simulation environment, comparison charts and graphs of the studied and simulated algorithms on various parameters of quality of service.

Chapter 5 describes the conclusion of the report and future work.

CHAPTER 2

LITERATURE REVIEWED

The following sections describe the literature background for the proposed problem statement given in chapter 1. Here, various authors had proposed their approaches to solve the various issues related to proposed problem statement related to our problem statement till now. We have categories the literature into two different sections: Scalable load balancing algorithms and Energy efficient load balancing algorithms.

2.1 Scalable load balancing algorithms:

Title: Analysis of Load Balancing Techniques in Cloud Computing

Author: Amandeep Kaur Sidhu, Supriya Kinger

Abstract: Amandeep et al. [4] has presented a concept of Cloud Computing along with research challenges in load balancing. Following load balancing algorithms are currently prevalent in clouds:

- ***Round Robin:*** In this algorithm, the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same.

- ***Connection Mechanism:*** Load balancing algorithm can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it, and decreases the number when connection finishes or timeout happens.

- ***Randomized:*** Randomized algorithm is of type static in nature. In this algorithm a process can be handled by a particular node n with a probability p. The process allocation order is maintained for each processor independent of allocation from remote processor. This algorithm works well in case of processes are of equal loaded. However, problem arises when loads are of different computational complexities. Randomized algorithm does not maintain deterministic approach. It works well when Round Robin algorithm generates overhead for process queue.

- ***Equally Spread Current Execution Algorithm:*** Equally spread current execution algorithm process handle with priorities. it distribute the load randomly by checking the size and transfer the load to that virtual machine which is lightly loaded or handle that task easy and take less time , and give maximize throughput. It is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines.

- ***Throttled Load Balancing Algorithm:*** Throttled is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is given by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

- ***A Task Scheduling Algorithm Based on Load Balancing:*** Y. Fang et al. discussed a two-level task scheduling mechanism based on load balancing to meet dynamic requirements of users and obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment.

- **Min-Min Algorithm:** It begins with a set of all unassigned tasks. First of all, minimum completion time for all tasks is found. Then among these minimum times the minimum value is selected which is the minimum time among all the tasks on any resources. Then according to that minimum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines.

- **Max-Min Algorithm**: Max-Min is almost same as the min-min algorithm except the following: after finding out minimum execution times, the maximum value is selected which is the maximum time among all the tasks on any resources. Then according to that maximum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines.

Title: A comparative study into distributed load balancing algorithms for cloud computing

Author: Randles, Martin, David Lamb, and A. Taleb-Bendiab

Abstract: Randles et al [7] has comparatively analyzed the distributed load balancing algorithms in cloud computing. Authors have compared the following distributed load balancing algorithms:

- *Honeybee ForagingBehavior*
- *Biased RandomSampling*
- *ActiveClustering*

Honeybee Foraging Behavior: This algorithm inspired by behavior of honeybees foraging and harvesting food. This approach is employed as a searching technique. In honeybee load balancing approach, set of servers are divided into virtual servers. Each virtual server is serving a virtual service request queue. To measure the bee's quality, cost of serving the request is calculated which gives theprofit.

Biased Random Sampling: in this approach, load of a server is measured by its connectivity in a virtual graph. Initially a network is created with virtual nodes that representeachserver.Degreeofeachservernodeismappedtoavailableresources.

Number of incoming edges gets connected with randomly selected nodes. Through this edge dynamics load allocation is required for load balancing. When a node process a new task, it deletes an inward edge, represent reduction in tasks. Adversely, when a node completes its task, a new incoming edge is added. The process of increment and decrement is performed via Random Sampling. During sampling, at each step node select its one of neighbor randomly.

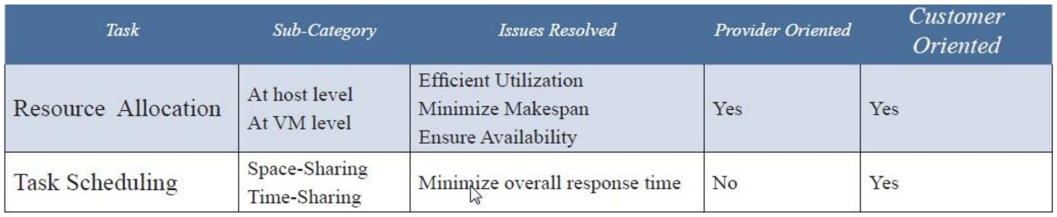
Active Clustering: It is self-aggregation algorithm to reconstruct the network. This approach works on the principle of similarity group. Active clustering consists of following iteration:

- At any time (random), a node acts as an initiator and select randomly different type of nodes from its neighbors.
- The matchmaker node leads to a link to be created between one matchmaker nodes.
- The matchmaker deletes that links.

Title: A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment

Authors: MayankaKatyal, Atul Mishra

Abstract: Mayanka Katyal et al [8] have presented various load balancing schemes in different cloud environment based on requirements specified in Service Level Agreement (SLA). Authours took into account two major tasks for load balancing, one is the resource provisioning or resource allocation and other is task scheduling in distributed environment. Table 2.gives the comparison of resource allocation and task scheduling and specifies the issues resolved by each technique of load balancing.



Task	*Sub-Category*	*Issues Resolved*	*Provider Oriented*	*Customer Oriented*
Resource Allocation	At host level At VM level	Efficient Utilization Minimize Makespan Ensure Availability	Yes	Yes
Task Scheduling	Space-Sharing Time-Sharing	Minimize overall response time	No	Yes

Table 1: Comparison between Resource Allocation and Task Scheduling

Based on resource provisioning and scheduling, four cases can be examined under different performance criteria so as to get efficient load balancing scheme.

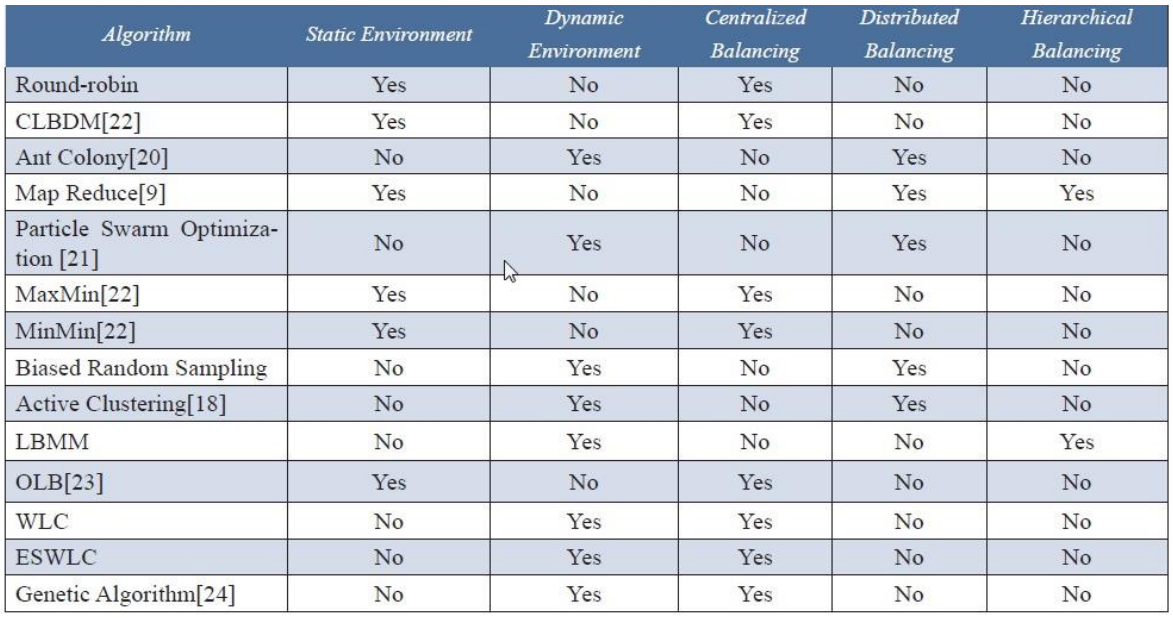
Case 1: Hosts and VMs, both are provisioned in space sharing manner.

Case 2: Hosts and VMs, both are provisioned to VMs and tasks respectively in time sharing manner.

Case 3: Hosts are provisioned to VMs in space sharing manner and VMs are provisioned to tasks in time sharing manner.

Case 4: Hosts are provisioned to VMs in time sharing manner and VMs are provisioned to tasks in space sharing manner.

Further they propose that in distributed scenario, failure intensity of a node is not neglected. Hence, the system is fault tolerant and balanced as well as no single node is overloaded to make load balancing decision. Comparison of different static and dynamic load balancing algorithms is given in Table 3. It also compares them on the basis of spatial distribution of nodes.



Algorithm	*Static Environment*	*Dynamic Environment*	*Centralized Balancing*	*Distributed Balancing*	*Hierarchical Balancing*
Round-robin	Yes	No	Yes	No	No
CLBDM[22]	Yes	No	Yes	No	No
Ant Colony[20]	No	Yes	No	Yes	No
Map Reduce[9]	Yes	No	No	Yes	Yes
Particle Swarm Optimization [21]	No	Yes	No	Yes	No
MaxMin[22]	Yes	No	Yes	No	No
MinMin[22]	Yes	No	Yes	No	No
Biased Random Sampling	No	Yes	No	Yes	No
Active Clustering[18]	No	Yes	No	Yes	No
LBMM	No	Yes	No	No	Yes
OLB[23]	Yes	No	Yes	No	No
WLC	No	Yes	Yes	No	No
ESWLC	No	Yes	Yes	No	No
Genetic Algorithm[24]	No	Yes	Yes	No	No

Table 2: Comparison Table of LBA in Cloud Computing Environment

In this paper, they discussed various load balancing schemes, each having some pros and cons. On one hand static load balancing scheme provide easiest simulation and monitoring of environment but fail to model heterogeneous nature of cloud. On the other hand, dynamic load balancing algorithm are difficult to simulate but are best suited in heterogeneous environment of cloud computing. Also the level at node which implements this static and dynamic algorithm plays a vital role in deciding the effectiveness of algorithm. Unlike centralized algorithm, distributed nature of algorithm provides better fault tolerance but requires higher degree of replication and on the other hand, hierarchical algorithm divide the load at different levels of hierarchy with upper level nodes requesting for services of lower level nodes in balanced manner. Hence, dynamic load balancing techniques in distributed or hierarchical environment provide better performance.

Title: Dynamically Scaling Applications in the Cloud

Author: Luis M. Vaquero, Luis Rodero-Merino, RajkumarBuyya

Abstract: Luis et al[9] have proposed the most notable initiatives towards whole application scalability in cloud environments. Having several servers and the mechanisms to distribute load among nodes is a definitive step towards scaling a cloud application. However, according to them, Network scalability is the often neglected element of the datacentre infrastructure which needs to be considered towards complete application scalability. Cloud applications should be able to request not only virtual servers at multiple points in the network, but also bandwidth-provisioned network pipes and other network resources to interconnect them. Clouds that offer simple virtual hardware infrastructure such as VMs and networks are usually denoted IaaS Clouds. A different abstraction level is given by PaaS clouds which supply a container-like environment where users deploy their applications as software components. PaaS clouds provide sets of “online libraries” and services for supporting application design, implementation and maintenance. Despite being somewhat less flexible than IaaS clouds, PaaS clouds are becoming important elements for building applications in a faster manner and many important IT players such as Google and Microsoft have developed new PaaS clouds systems such as Google App Engine and Microsoft Azure. Due to their importance this document also discusses scalability in PaaS clouds at two different levels: container level and database level. Figure 1 provides an overview of the mechanisms handy to accomplish the goal of whole application scalability

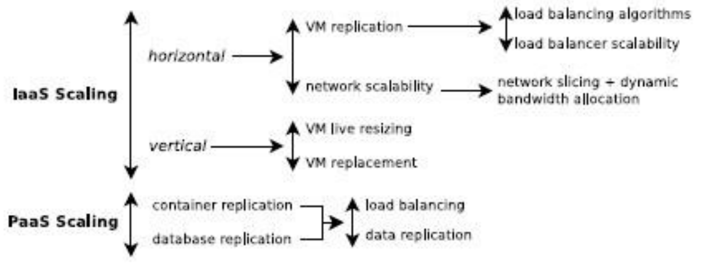


Figure 9: Summary of available mechanisms for holistic application scalability

Title: Load Balancing Method for Infrastructure as a Service (IaaS) In Cloud Computing

Authors: Nikhil S. Dharane, Abhijeet S. Kulkarni, Akshay U. Bhawthankar, A.A. Deshmukh

Abstract: Nikhil et al [10] focuses on the IaaS cloud, its optimization and further the authors give their proposed model for load balancing in IaaS. IaaS cloud is usually provided to users in the form of Virtual Machines (VMs), such as Amazon EC2. In an IaaS cloud, users can apply VMs on-demand to deploy and run their applications also provide services to their clients which will be helpful to clients. IaaS has major issues like resource management, network infrastructure, virtualization, data management etc. IaaS provides benefits like: scalability, QoS, reduction in overheads, cost effectiveness. Different types of resources like physical and logical are provided in IaaS. There are physical servers with a large number of virtual machines. These virtual machines are hosted with many heterogeneous applications. In order to optimize the utilization of computing resources and also saving energy consumption of cloud data centers, the applications running on the virtual machines will be migrated either to the same server or to another physical or virtual server. Identifying when it is best to migrate an application in a virtual machine has a direct impact on resource optimization. Performance optimization can be best achieved by an efficiently monitoring the utilization of computing resources. So, this calls for the need of comprehensive intelligent monitoring agent to analyse the performances of virtual machines.

They further proposed a model that had two cases one for VM starting and other for load is increasing or falling below threshold. Fig 1 shows system architecture. First we get the next several hours prediction load of the starting VM. Then, we select n hosts that have lower load. Then, one suitable host will be select from these n hosts for the VM running on. The principle for choose this host is that, if this VM running on the host, the load-balancing factor will be the minimum in next several hours.

For the second case which the load of some hosts exceeds or falls below the threshold, several hosts with lowest load and highest load will be selected, and some suitable VMs on high load hosts would be chosen to migration to the hosts with lower load. In the extreme case, the load of every host is below the threshold, we need migrate the VMs of some hosts and shutdown these hosts. Conversely, if the load of every host is higher than the threshold, new hosts would start.

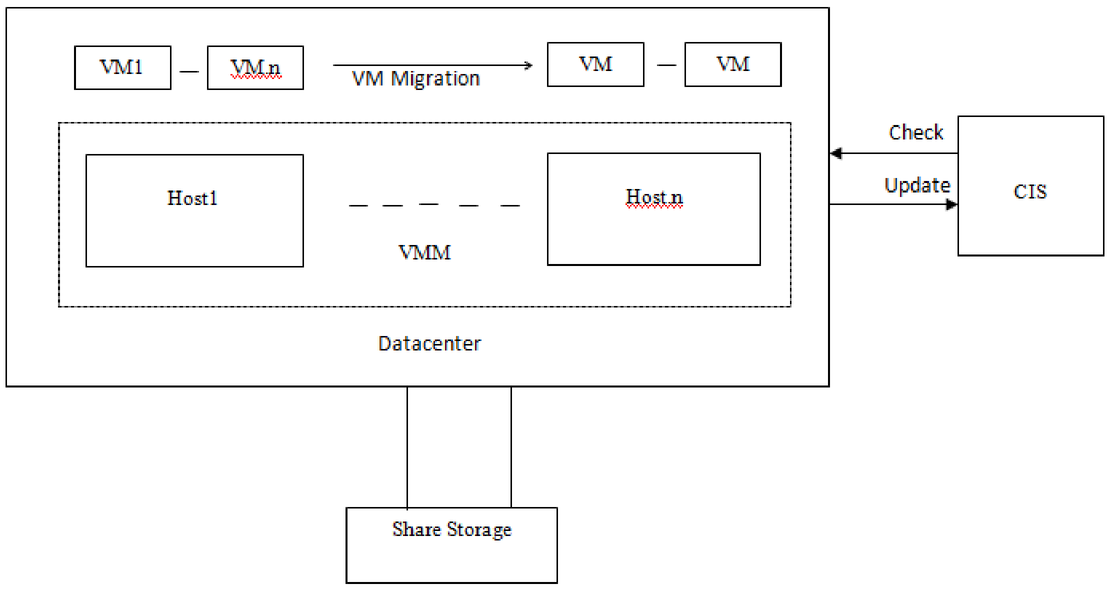
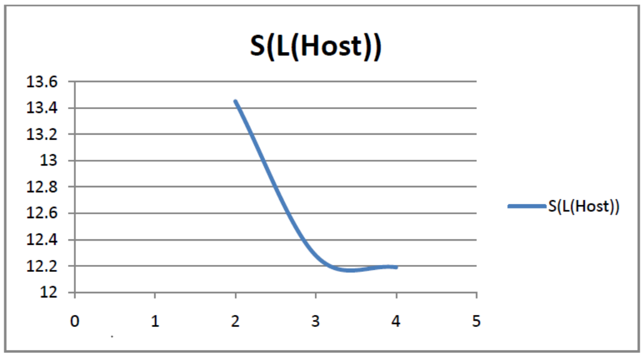


Figure 10 a: System Architecture of proposed algorithm

This graph shows that when number of hosts increases then load balancing factor decreases. So load is perfectly balanced between all available hosts.



x axis = No of hosts y axis = load balancing factor (CPU utilization)

Figure 10 b: Number of host v/s load balancing factors

Title: Load balancing strategy of cloud computing based on artificial bee algorithm

Author: Yao, Jing, and Ju-hou He

Abstract: Yao et al. [11] have proposed an improved Artificial Bee Colony algorithm. They have experimentally represented that ABC based load balancing algorithm outperform the basic ABC algorithm [24, 25]. Authors have said that previous load balancing algorithms consider only lightly loaded node and execute a lot of requests e.g. newly arrived request and requests coming from heavily loaded nodes. This leads to load imbalance again.

2.2 Energy Efficient Load Balancing Algorithms:

Title: Energy Aware Load Balancing In Cloud Computing Using Virtual Machines

Author: Maunika M. Ramani, Prof. Mohammed H. Bohara

Abstract: Maunika et al[12] in their proposed work considered the situation of over-utilization, under-utilization using resource utilization threshold and control temperature of the host using temperature threshold. Their goal is to save maximum energy of the data centre. In this approach, they have identified the dynamic threshold value of resource utilization and define temperature threshold. It reduces the consumption of maximum resources, control the temperature of the processor and maximum minimize the energy consumption.

They have proposed a technique, which base on threshold values of resource utilization, temperature threshold values and VM consolidation. Resource utilization threshold values are calculated dynamically. Using it has improved the resource utilization so balance the workload in a better way. Temperature of CPU is calculated using a lumped thermal model which controls the heat of CPU. Using these thresholds identifies over loaded or heated hosts and under loaded or heated host. After that, using some **ThaS** steps and VM consolidation to achieve the final goal. Figure 11 defines the workflow diagram of our propose work.

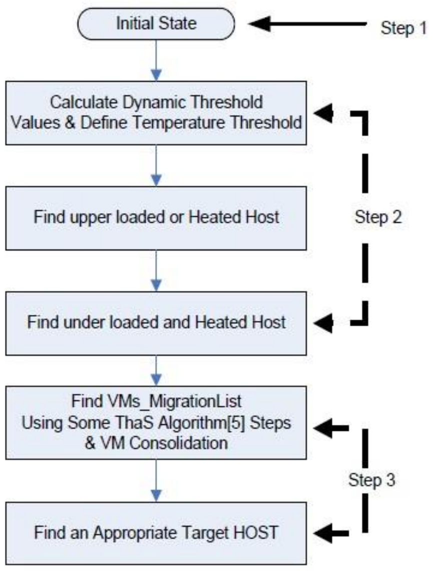

Initial State
Step 1
Calculate Dynamic Threshold Values & Define Temperature Threshold
Find upper loaded or Heated Host
Step 2
Find under loaded and Heated Host
Find VMs_MigrationList Using Some ThaS Algorithm[5] Steps & VM Consolidation
Step 3
Find an Appropriate Target HOST

Figure 11: Flow of proposed work

Steps of the propose work which defines the overall process of the work:

Step 1: Figure shows the of the initial state of the datacentre .In data centre have different host with the workload. There are different types of load in the host machine as CPU load, storage load, memory load, network related load, etc. Temperature of host also calculated using lumped thermal model. Initial state defines as in figure 12.

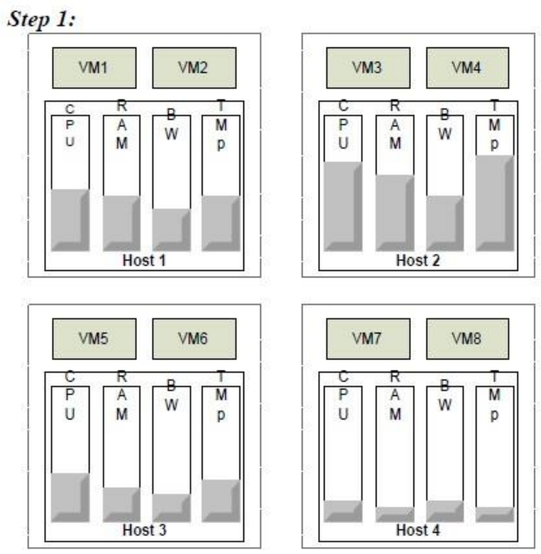


Figure 12: Initial state of datacenter

Step 2: At the second stage calculate dynamic thresholds of resource utilization and define temperature thresholds. Resource utilization threshold values identify over/under loaded host machines. And, using a temperature threshold identifies high heated and low heated host machines to control the temperature of resources and save the unnecessary usage of energy. In figure13 define these hosts.

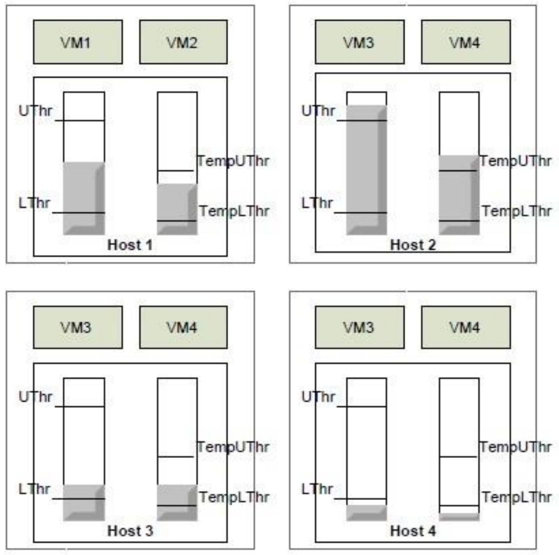


Figure 13: Identify under/overloaded and heated hosts.

Step 3:

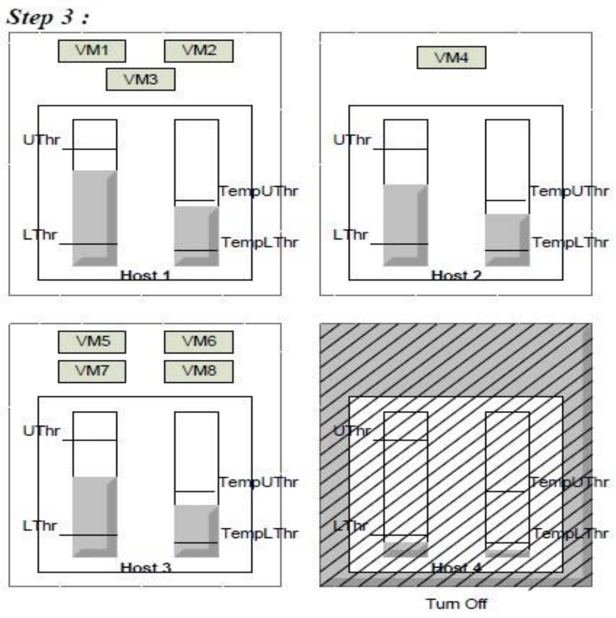


Figure 14: VM migrates on appropriate target hosts.

Title: Energy Efficient Resource Management in Virtualized Cloud Data Centers

Author: Anton Beloglazov and Rajkumar Buyya

Abstarct: Anton et al[13] proposed a decentralized architecture of the energy aware resource management system for Cloud data centers. They have defined the problem of minimizing the energy consumption while meeting QoS requirements and stated the requirements for VM allocation policies. Moreover, they have proposed three stages of continuous optimization of VM placement and presented heuristics for a simplified version of the first stage. The heuristics have been evaluated by simulation using the extended CloudSim toolkit. One of the heuristics leads to significant reduction of the energy consumption by a Cloud data center – by 83% in comparison to a non power aware system and by 66% in comparison to a system that applies only DVFS technique but does not adapt allocation of VMs in run-time. Moreover, MM policy enables flexible adjustment of SLA by setting appropriate values of the utilization thresholds: SLA can be relaxed leading to further improvement of energy consumption. The policy supports heterogeneity of both the hardware and VMs and does not require any knowledge about particular applications running on the VMs. The policy is independent of the workload type.

SYSTEM ARCHITECTURE

In this work the underlying infrastructure is represented by a large-scale Cloud data center comprising *n* heterogeneous physical nodes. Each node has a CPU, which can be multicore, with performance defined in Millions Instructions Per Second (MIPS).Users submit requests for provisioning of *m* heterogeneous VMs with resource requirements defined in MIPS, amount of RAM and network bandwidth. SLA violation occurs when a VM cannot get the requested amount of resource, which may happen due to VM consolidation. As shown in Figure 15, the system operation consists of the following steps:

1) New requests for VM provisioning. Users submit requests for provisioning of VMs.

2) Dispatching requests for VM provisioning. The dispatcher distributes requests among global managers.

3) Intercommunication between global managers. The global managers exchange information about utilization of resources and VMs that have to be allocated.

4) Data about utilization of resources and VMs chosen to migrate. The local managers propagate information about resource utilization and VMs chosen to migrate to the global managers.

5) Migration commands. The global managers issue VM migration commands in order to optimize current allocation.

6) Commands for VM resizing and adjusting of power states. The local managers monitor their host nodes and issue commands for VM resizing and changes in power states of nodes.

7) VM resizing, scheduling and migration actions. According to the received commands, VMM performs actual resizing and migration of VMs as well as resource scheduling.

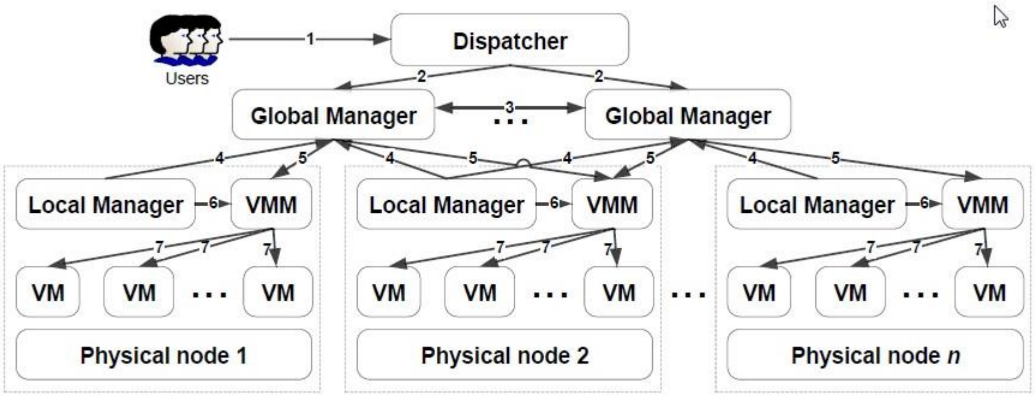


Figure 15: The system architecture

The simulation results are presented in Table 3. The results show that dynamic reallocation of VMs according to current utilization of CPU can bring higher energy savings comparing to static allocation policies. MM policy allows to achieve the best energy savings: by 83%, 66% and 23% less energy consumption relatively to NPA, DVFS and ST policies respectively with thresholds 30-70% and ensuring percentage of SLA violations of 1.1%; and by 87%, 74% and 43% with thresholds 50-90% and 6.7% of SLA violations. MM policy leads to more than 10 times fewer VM migrations than ST. The results show the flexibility of the algorithm, as the thresholds can be adjusted according to SLA requirements. Strict SLA (1.11%) allow achievement of the energy consumption of 1.48 KWh. However, if SLA are relaxed (6.69%), the energy consumption is further reduced to 1.14 KWh.

Policy	Energy	SLA	Migr.	Avg. SLA
NPA	9.15 KWh	-	-	-
DVFS	4.40 KWh	-	-	-
ST 50%	2.03 KWh	5.41%	35 226	81%
ST 60%	1.50 KWh	9.04%	34 231	89%
MM 30-70%	1.48 KWh	1.11%	3 359	56%
MM 40-80%	1.27 KWh	2.75%	3 241	65%
MM 50-90%	1.14 KWh	6.69%	3 120	76%

Table 3: Simulation Results

Title: Existing Load Balancing Techniques In Cloud Computing: A Systematic Review

Author: Nidhi Jain Kansal, Inderveer Chana

Abstract: Nidhi et al. [14] discussed the existing load balancing techniques that are currently prevalent in clouds and did the comparative study based on the parameters as visible in the table

1. VectorDot: VectorDot uses dot product to distinguish nodes based on the item requirements and helps in removing overloads on servers, switches and storage nodes.

2. **CARTON:** a mechanism CARTON for cloud control that unifies the use of LB and DRL. LB (Load Balancing) is used to equally distribute the jobs to different servers so that the associated costs can be minimized and DRL (Distributed Rate Limiting) is used to make sure that the resources are distributed in a way to keep a fair resource allocation.

3. **Compare and Balance:** A distributed load balancing algorithm COMPARE AND BALANCE is proposed that is based on sampling and reaches equilibrium very fast. This algorithm assures that the migration of VMs is always from high-cost physical hosts to low-cost host but assumes that each physical host has enough memory which is a weak assumption.

4. **Event-driven:** This algorithm after receiving capacity events as input, analyses its components in context of the resources and the global state of the game session, thereby generating the game session load balancing actions. It is capable of scaling up and down a game session on multiple resources according to the variable user load but has occasional QoS breaches.

5. **Scheduling strategy on LB of VM resources:** This strategy achieves the best load balancing and reduced dynamic migration by using a genetic algorithm. It helps in resolving the issue of load imbalance and high cost of migration thus achieving better resource utilization.

6. **CLBVM:** This policy improves the overall performance of the system but does not consider the systems that are fault-tolerant.

7. **LBVS:** Storage virtualization is achieved using an architecture that is three-layered and load balancing is achieved using two load balancing modules. It helps in improving the efficiency of concurrent access by using replica balancing further reducing the response time and enhancing the capacity of disaster recovery. This strategy also helps in improving the use rate of storage resource, flexibility and robustness of the system.

8. **Task Scheduling based on LB:** It achieves load balancing by first map-ping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment.

9. **Honeybee Foraging Behaviour:** nature-inspired algorithm for self-organization. It achieves global load balancing through local server actions. Performance of the system is enhanced with increased sys-tem diversity but throughput is not increased with an increase in system size. It is best suited for the conditions where the diverse population of service types is required.

10. **Biased Random Sampling:** distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system.

11. Active Clustering: a self-aggregation load balancing technique that is a self-aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring.

12. **ACCLB:** a load balancing mechanism based on ant colony and complex network theory (ACCLB) in an open cloud computing federation. It uses small-world and scale-free characteristics of a complex network to achieve better load balancing.

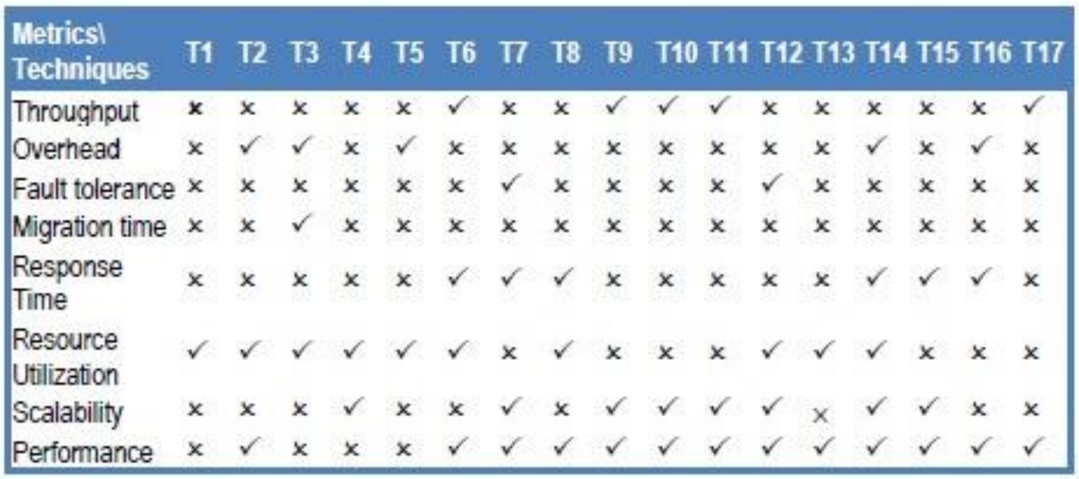
13. **(OLB + LBMM):** combination of two. OLB scheduling algorithm, keeps every node in working state to achieve the goal of load balance and LBMM scheduling algorithm is utilized to minimize the execution time of each task on the node thereby minimizing the overall completion time.

14. **Decentralized content aware:** It uses a parameter named as USP to specify the unique and special property of the requests as well as computing nodes. USP helps the scheduler to decide the best suitable node for processing the requests.

15. **Server-based LB for Internet distributed services:** a new server-based load balancing policy for web servers which are distributed all over the world. It helps in reducing the service response times by using a protocol that limits the redirection of requests to the closest remote servers without overloading them.

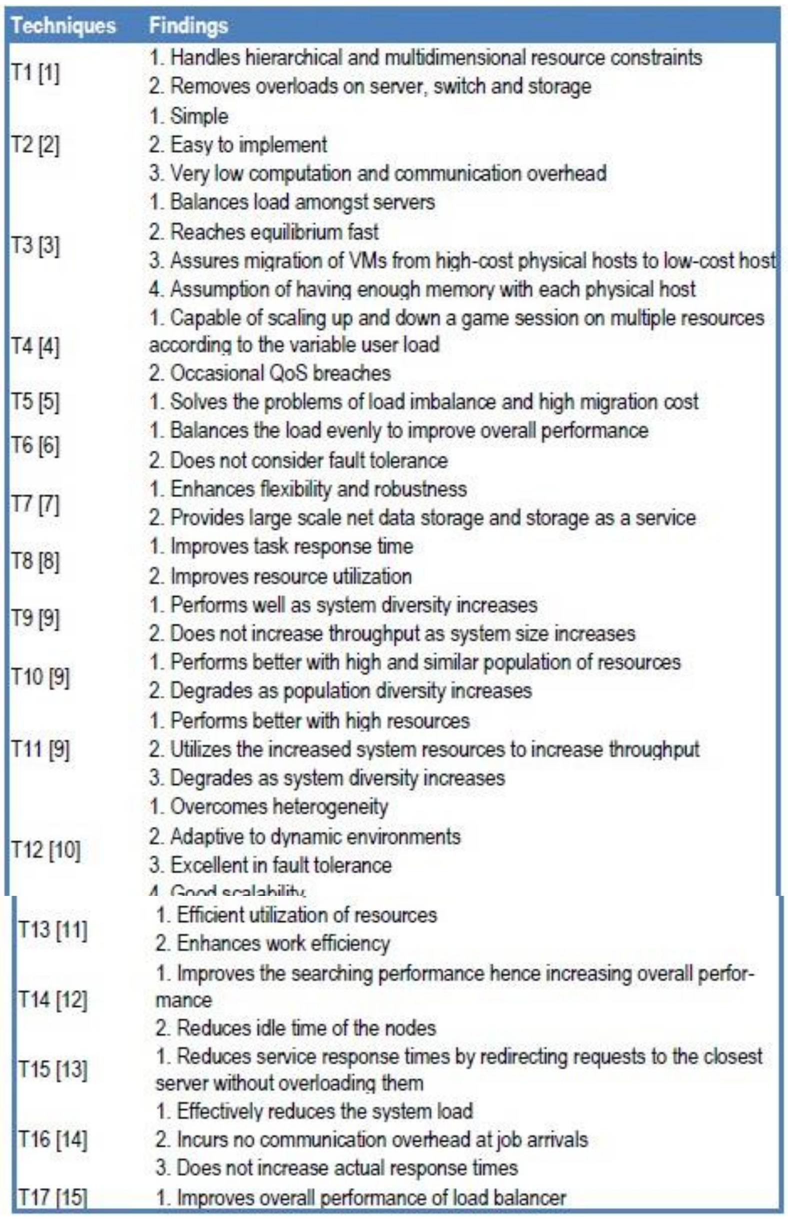
16. J**oin-Idle-Queue:** This algorithm provides large-scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers for the availability of idle processors at each dispatcher and then, assigning jobs to processors to reduce average queue length at each processor.

17. Lock-free multiprocessing solution for LB: a lock-free multiprocessing load balancing solution that avoids the use of shared memory in contrast to other multiprocessing load balancing solutions which use shared memory and lock to maintain a user session. It is achieved by modifying Linux kernel.



Metrics\ Techniques	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
Throughput	×	×	×	×	×	✓	×	×	✓	✓	✓	×	×	×	×	×	✓
Overhead	×	✓	✓	×	✓	×	×	×	×	×	×	×	×	✓	×	✓	×
Fault tolerance	×	×	×	×	×	×	✓	×	×	×	×	✓	×	×	×	×	×
Migration time	×	×	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Response Time	×	×	×	×	×	✓	✓	✓	×	×	×	×	×	✓	✓	✓	×
Resource Utilization	✓	✓	✓	✓	✓	✓	×	✓	×	×	×	✓	✓	✓	×	×	×
Scalability	×	×	×	✓	×	×	✓	×	✓	✓	✓	✓	×	✓	✓	×	×
Performance	×	✓	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 4: Comparison of LB techniques based on various metrics



Techniques	Findings
T1 [1]	1. Handles hierarchical and multidimensional resource constraints 2. Removes overloads on server, switch and storage
T2 [2]	1. Simple 2. Easy to implement 3. Very low computation and communication overhead
T3 [3]	1. Balances load amongst servers 2. Reaches equilibrium fast 3. Assures migration of VMs from high-cost physical hosts to low-cost host 4. Assumption of having enough memory with each physical host
T4 [4]	1. Capable of scaling up and down a game session on multiple resources according to the variable user load 2. Occasional QoS breaches
T5 [5]	1. Solves the problems of load imbalance and high migration cost
T6 [6]	1. Balances the load evenly to improve overall performance 2. Does not consider fault tolerance
T7 [7]	1. Enhances flexibility and robustness 2. Provides large scale net data storage and storage as a service
T8 [8]	1. Improves task response time 2. Improves resource utilization
T9 [9]	1. Performs well as system diversity increases 2. Does not increase throughput as system size increases
T10 [9]	1. Performs better with high and similar population of resources 2. Degrades as population diversity increases
T11 [9]	1. Performs better with high resources 2. Utilizes the increased system resources to increase throughput 3. Degrades as system diversity increases
T12 [10]	1. Overcomes heterogeneity 2. Adaptive to dynamic environments 3. Excellent in fault tolerance 4. Good scalability
T13 [11]	1. Efficient utilization of resources 2. Enhances work efficiency
T14 [12]	1. Improves the searching performance hence increasing overall perfor-mance 2. Reduces idle time of the nodes
T15 [13]	1. Reduces service response times by redirecting requests to the closest server without overloading them
T16 [14]	1. Effectively reduces the system load 2. Incurs no communication overhead at job arrivals 3. Does not increase actual response times
T17 [15]	1. Improves overall performance of load balancer

Table 5: Analysis of existing LB techniques

Title: Optimal Online Deterministic Algorithm and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers

Author: Anton Beloglazov and Rajkumar Buyya

Abstract: Anton et al [15] explained the rapid growth in demand for computational power driven by modern service applications combined with the shift to the Cloud computing model have led to the establishment of large-scale virtualized data centers. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) using live migration and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. However, the obligation of providing high quality of service to customers leads to the necessity in dealing with the energy-performance trade-off, as aggressive consolidation may lead to performance degradation. Due to the variability of workloads experienced by modern applications, the VM placement should be optimized continuously in an online manner.

Title: Cloud task scheduling based on load balancing ant colony optimization

Author: Li, Kun, GaochaoXu, Guangyu Zhao, Yushuang Dong, and Dan Wang

Abstract: Kun-Li et al. [16] have proposed a scheduling algorithm based on the Load Balancing Ant Colony Optimization (LBACO) which is an enhanced version of simple ACO algorithm[27,28]. Authors have tried to balance the load and minimize the response time of a tasks. Authors have simulated the approach using the Cloudsim simulator and compared it with FCFS and basic ACO algorithm.

2.3 Conclusion

In the literature review, some authors have proposed problems related to scalable load balancing approach and some authors have proposed energy efficient load balancing approach to solve the issues related to proposed problem statement. Both types of approaches have some pros and cons. So the applicability of the any approach depends on the scenario used by the authors. The main motive of literature review is to thoroughly analyze the research work related to proposed problem statement and design an approach to solve the issues related to proposed problem statement.

CHAPTER 3
PROPOSED WORK

3.1. Proposed approach

We have proposed a distributed load balancing algorithms in cloud environment. Keeping into the consideration the parameters of Quality of Service (QoS) which includes scalability, energy efficiency, fault tolerance, flexibility, response time, migration time of the resources. For this we have first studied already existing algorithms DVFS and IQR and simulated them with RR and FCFS. Results are being noted down and comparative studies of those graphs have been done to identify which parameters lack behind in the current scenario.

3.2. Dynamic Voltage and Frequency Scaling and its Simulation

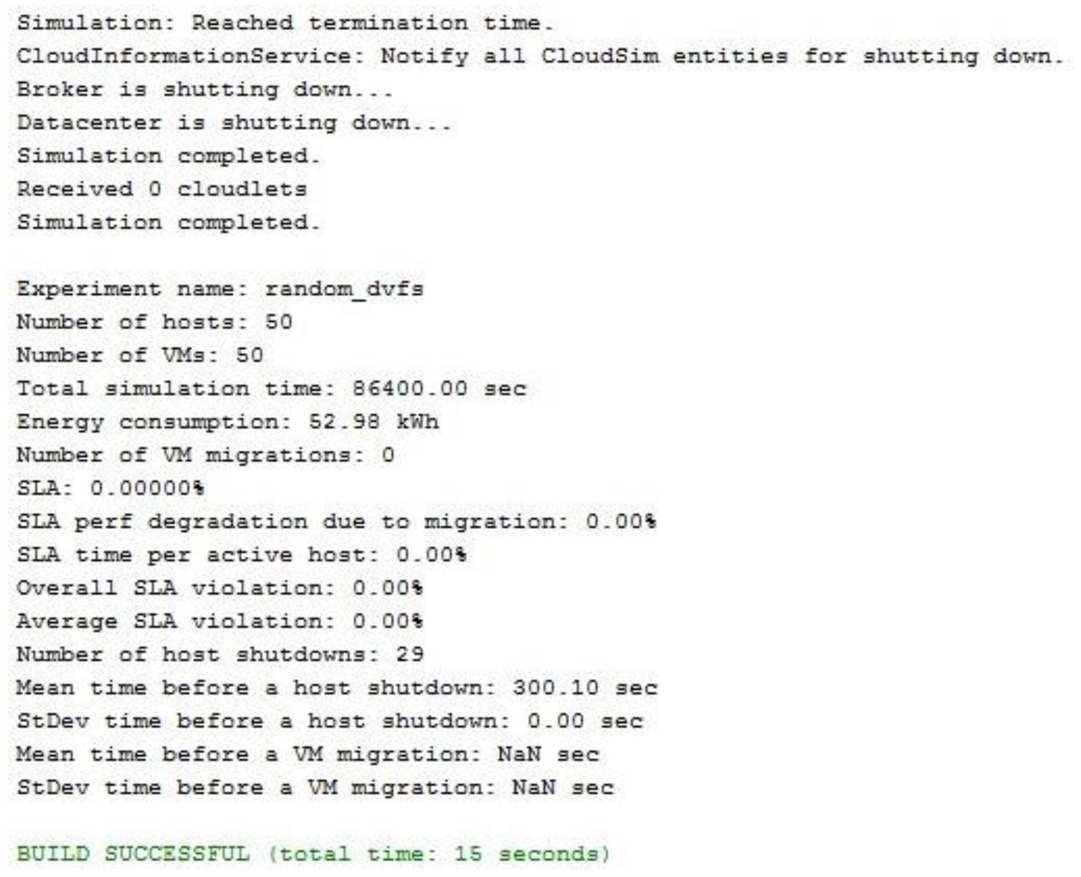
DVFS is the simulation of a heterogeneous power aware data center that only applied DVFS, but no dynamic optimization of the VM allocation. The adjustment of the hosts' power consumption according to their CPU utilization is happening in the PowerDatacenter class. The remaining configuration parameters are in the Constants and Random Constants classes. The use of DVFS directly affects the performance of the CPU capacity (and hosts), which are subject to regular changes during simulations. This also involves changing the way the simulator handles the placement and management of virtual machines. For example, if the system decides to reduce the frequency, the sum of capacities of all virtual machines (VMs) may temporarily exceed the maximum capacity of the host. In this case, the size of each VM must temporarily be adjusted downward in proportion to the capacity of the new host. The same situation occurs when one or more new virtual machines have to be added to a host already in use. If the sum of capacities of the virtual machines already running in the host plus the capacities of new virtual machines created exceeds the capacity

of the host, the size of all virtual machines has to be decreased before adding the new virtual machines to the host.
Each DVFS mode can be configured in different ways by setting specific parameters values.[18]

- Sampling rate: defines the minimum time interval between two frequency changes;
- Thresholds: The OnDemand and Conservative modes compare the current CPU load with predefined thresholds. By setting custom values, the user can adapt their behavior.

3.2.1 DVFS Simulation with FCFS

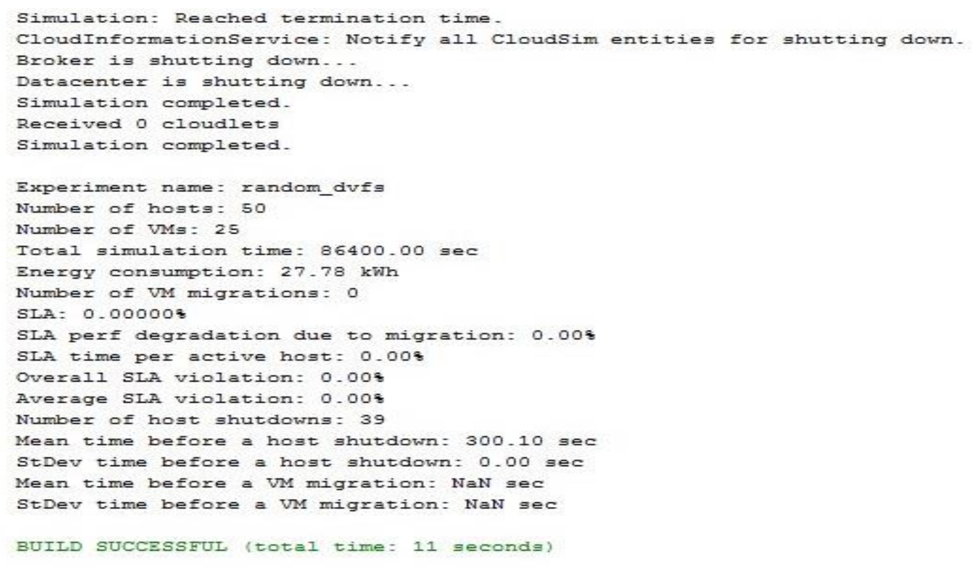
CASE 1: No. of Hosts: 50 and No. of VMs: 50



```
Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 0 cloudlets
Simulation completed.

Experiment name: random_dvfs
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 52.98 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 29
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec

BUILD SUCCESSFUL (total time: 15 seconds)
```

Figure 16: screenshot of dvfs simulation with fcfs case 1

CASE 2: No. of Hosts: 50 and No. of VMs: 25

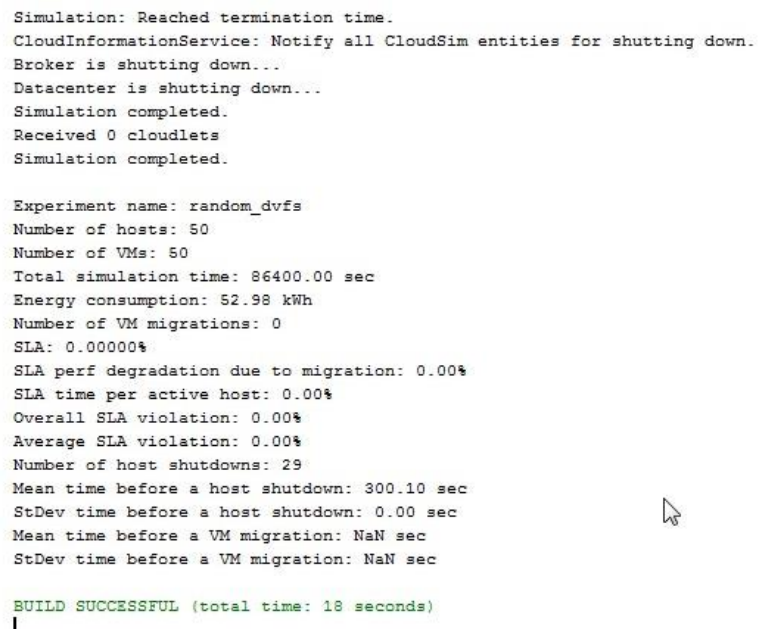


```
Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 0 cloudlets
Simulation completed.

Experiment name: random_dvfs
Number of hosts: 50
Number of VMs: 25
Total simulation time: 86400.00 sec
Energy consumption: 27.78 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 39
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec

BUILD SUCCESSFUL (total time: 11 seconds)
```

Figure 17: Screenshot of dvfs simulation with fcfs case 2

3.2.1 DVFS Simulation with RR

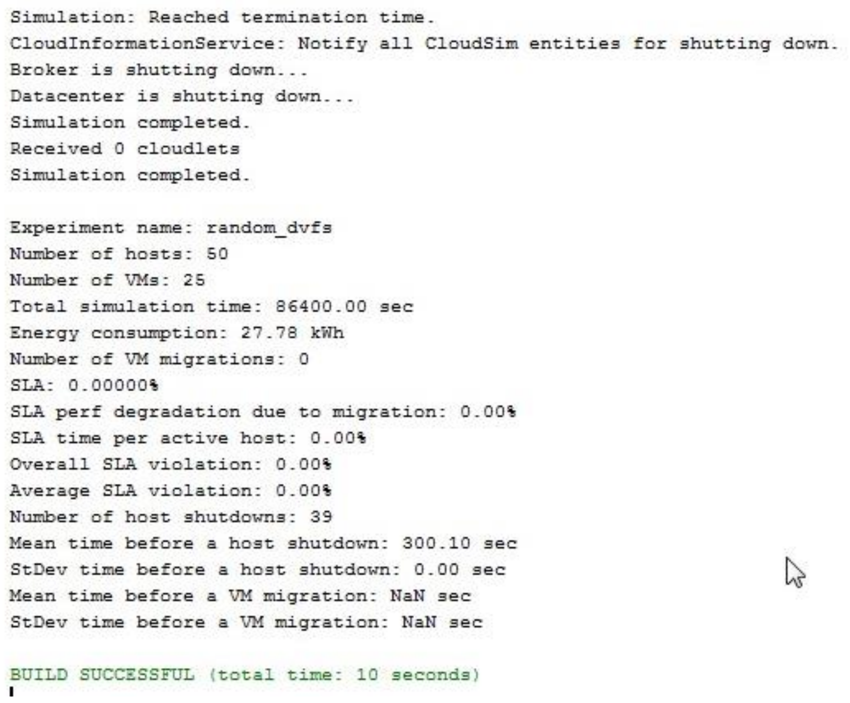
CASE 1: No. of Hosts: 50 and No. of VMs: 50



```
Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 0 cloudlets
Simulation completed.

Experiment name: random_dvfs
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 52.98 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 29
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec

BUILD SUCCESSFUL (total time: 18 seconds)
```

Figure 18: Screenshot of dvfs simulation with rr case 1

CASE2: No. of Hosts: 50 and No. of VMs: 25



```
Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 0 cloudlets
Simulation completed.

Experiment name: random_dvfs
Number of hosts: 50
Number of VMs: 25
Total simulation time: 86400.00 sec
Energy consumption: 27.78 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 39
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec

BUILD SUCCESSFUL (total time: 10 seconds)
```

Figure 19: Screenshot of dvfs simulation with rr case 2

3.3. Inter Quartile Range and its Simulation

The interquartile range (IQR) is an estimate of variability, based on dividing a data set into quartiles. It is the difference between the upper and lower quartile in a data set.

Steps for finding Interquartile Range:

1. Sort the data set in increasing order.
2. Find the median for the ordered set (Q2).
3. Divide the data set into two halves.
4. Find the median for the first half of the ordered data set (Lower Quartile Q1).
5. Find the median for the second half of the ordered data set (Upper Quartile Q3).
6. IQR = Upper Quartile – Lower Quartile.

Here data set defines set of the host utilization. We propose a method based on two threshold values, lower threshold and upper threshold. The median for the first half of the ordered data set (host utilization) is used to calculate the lower threshold value, while median of second half of the ordered data set (host utilization) is used to calculate the upper threshold value. This is shown in example as follows:

Let us assume the utilization of each host (in terms of percentage).

List of host utilization [23,65,10,75,50,84,15,30,90,12]

1. After sorting [10, 12, 15, 23, 30, 50, 65, 75, 84, 90]
2. Median = (30+50) / 2 =40
3. First half [10, 12, 15, 23, 30], Second half [50, 65, 75, 84, 90]
4. Median of the first half =15, which is used as lower threshold value.
5. Median of Second half = 75, which is used as upper threshold value.

Adaptive Heuristics for Dynamic VM Consolidation

Dynamic consolidation problems:

1. Determining when a host is considered as being overloaded
2. Determining when a host is considered as being underloaded
3. Selection of VMs that should be migrated from an overloaded host
4. Finding a new placement of VMs selected for migration from overloaded and underloaded hosts.

Algorithm 1[15]: VM placement Optimization

```
1. Input: hostList Output: migrationMap
2. foreach host in hostList do
3. if isHostOverloaded (host) then
4. vmsToMigrate.add(getVmsToMigrateFromOverloadedHost (host)
5. migrationMap.add(getNewVmPlacement (vmsToMigrate))
6. vmsToMigrate.clear()
7. foreach host in hostList do
8. if isHostUnderloaded (host) then
9. vmsToMigrate.add(host.getVmList( )
10. migrationMap.add(getNewVmPlacement (vmsToMigrate))
11. return migrationMap
```

Algorithm 2[16]: VM placement algo for Overloaded Host

```
1. For each VM in VMmigrationList1 {
2. For each Host in MostLikelyOverloadedHostList {
3. If (this host is suitable for VM) {
4. Calculate utlzAfterAllocation;
5. Calculate powerAfterAllocation;
6. If ((utlzAfterAllocation> prevUtlzOnThisHost)
&& (powerAfterAllocation < minpower))
7. targetHost = thisHost
8. }
9. }
10. Add (VM, Host) pair to MigrationMap
11. }
```

Algorithm 3[16]: VM placemen algo for Underloaded Host

```
1. For each VM in VMmigrationList2 {
2. For each Host in MostLikelyUnderloadedHostList {
```

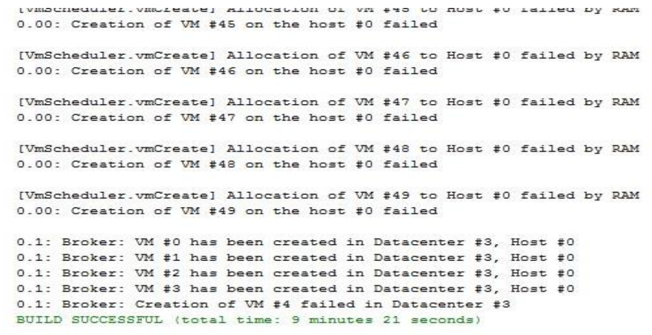
```
3. If (this host is suitable for VM) {
4. Calculate utlzAfterAllocation;
5. Calculate powerAfterAllocation;
6. If ((utlzAfterAllocation > prevUtlzOnThisHost)
&& (powerAfterAllocation < minpower)) {
7. targetHost = thisHost
8. }
9. }
10. }
11. Add (VM, Host) pair to MigrationMap
12. }
```

VM Selection is done on basis of one of the three:

- Minimum Migration Time (MMT)
- Random Choice (RC)
- Maximum Correlation (MC)

3.3.1. IQRrs Simulation with FCFS

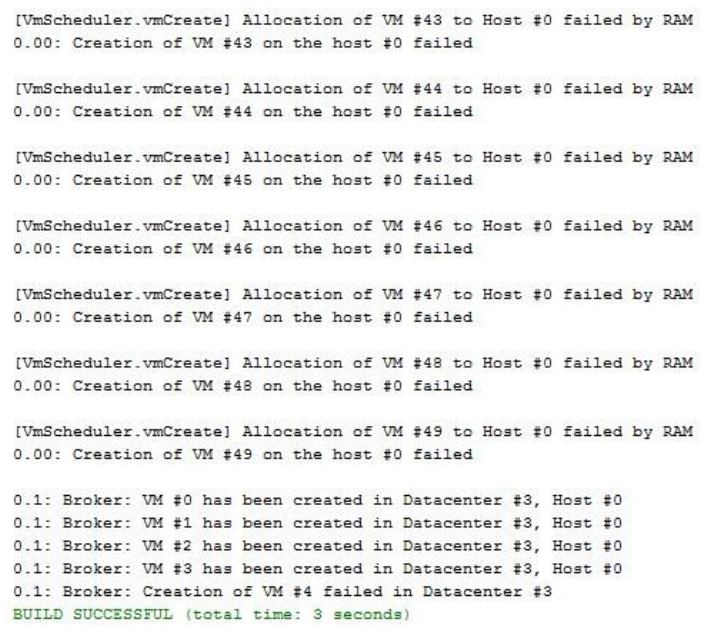
CASE1: No. of Hosts: 50 and No. of VMs: 50



```
[VmScheduler.vmCreate] Allocation of VM #45 to Host #0 failed by RAM
0.00: Creation of VM #45 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #46 to Host #0 failed by RAM
0.00: Creation of VM #46 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #47 to Host #0 failed by RAM
0.00: Creation of VM #47 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #48 to Host #0 failed by RAM
0.00: Creation of VM #48 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #49 to Host #0 failed by RAM
0.00: Creation of VM #49 on the host #0 failed

0.1: Broker: VM #0 has been created in Datacenter #3, Host #0
0.1: Broker: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker: VM #2 has been created in Datacenter #3, Host #0
0.1: Broker: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker: Creation of VM #4 failed in Datacenter #3
BUILD SUCCESSFUL (total time: 9 minutes 21 seconds)
```

Figure 20: Screenshot of IQRrs simulation with fcfs case1

CASE2: No. of Hosts: 25 and No. of VMs: 50

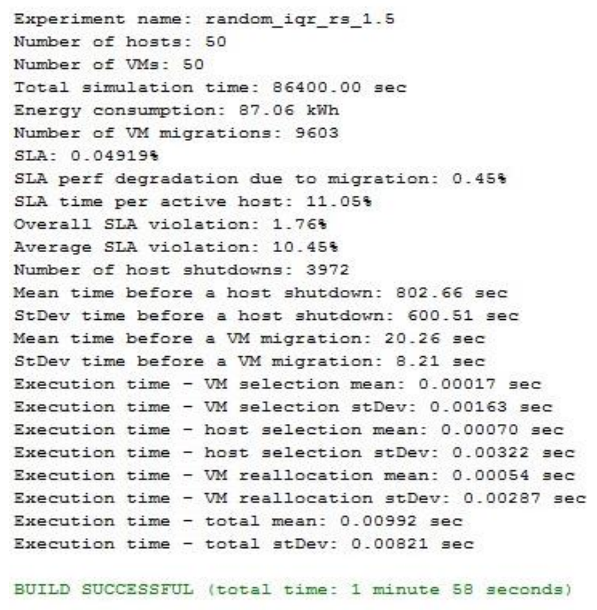


```
[VmScheduler.vmCreate] Allocation of VM #43 to Host #0 failed by RAM
0.00: Creation of VM #43 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #44 to Host #0 failed by RAM
0.00: Creation of VM #44 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #45 to Host #0 failed by RAM
0.00: Creation of VM #45 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #46 to Host #0 failed by RAM
0.00: Creation of VM #46 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #47 to Host #0 failed by RAM
0.00: Creation of VM #47 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #48 to Host #0 failed by RAM
0.00: Creation of VM #48 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #49 to Host #0 failed by RAM
0.00: Creation of VM #49 on the host #0 failed

0.1: Broker: VM #0 has been created in Datacenter #3, Host #0
0.1: Broker: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker: VM #2 has been created in Datacenter #3, Host #0
0.1: Broker: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker: Creation of VM #4 failed in Datacenter #3
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figure 21: Screenshot of IQRrs simulation with fcfs case2

3.3.2. IQRrs Simulation with RR

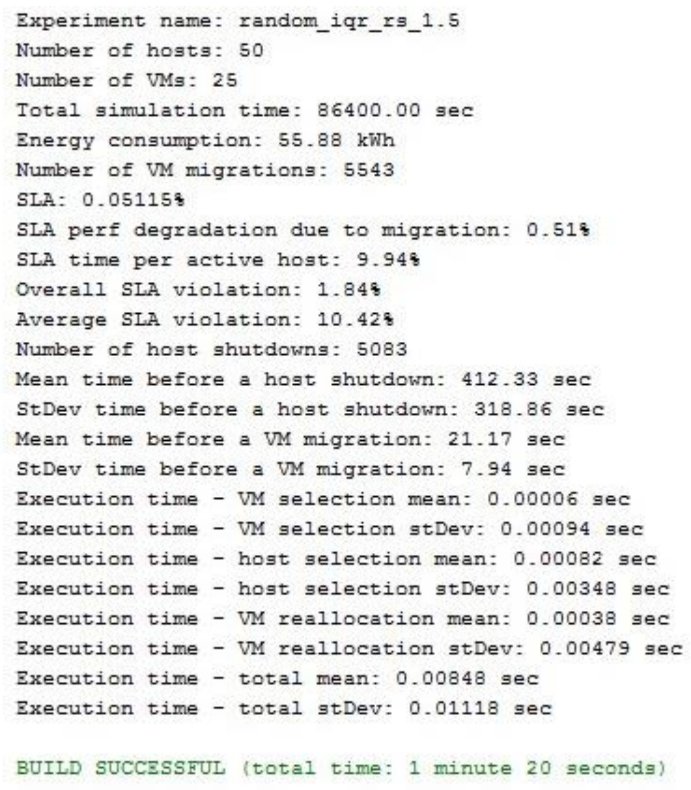
CASE1: No. of Hosts: 50 and No. of VMs: 50



```
Experiment name: random_iqr_rs_1.5
Number of hosts: 50
Number of VMs: 50
Total simulation time: 86400.00 sec
Energy consumption: 87.06 kWh
Number of VM migrations: 9603
SLA: 0.04919%
SLA perf degradation due to migration: 0.45%
SLA time per active host: 11.05%
Overall SLA violation: 1.76%
Average SLA violation: 10.45%
Number of host shutdowns: 3972
Mean time before a host shutdown: 802.66 sec
StDev time before a host shutdown: 600.51 sec
Mean time before a VM migration: 20.26 sec
StDev time before a VM migration: 8.21 sec
Execution time - VM selection mean: 0.00017 sec
Execution time - VM selection stDev: 0.00163 sec
Execution time - host selection mean: 0.00070 sec
Execution time - host selection stDev: 0.00322 sec
Execution time - VM reallocation mean: 0.00054 sec
Execution time - VM reallocation stDev: 0.00287 sec
Execution time - total mean: 0.00992 sec
Execution time - total stDev: 0.00821 sec

BUILD SUCCESSFUL (total time: 1 minute 58 seconds)
```

Figure 22: Screenshot of IQRrs simulation with rr case1

CASE2: No. of Hosts: 50 and No. of VMs: 25

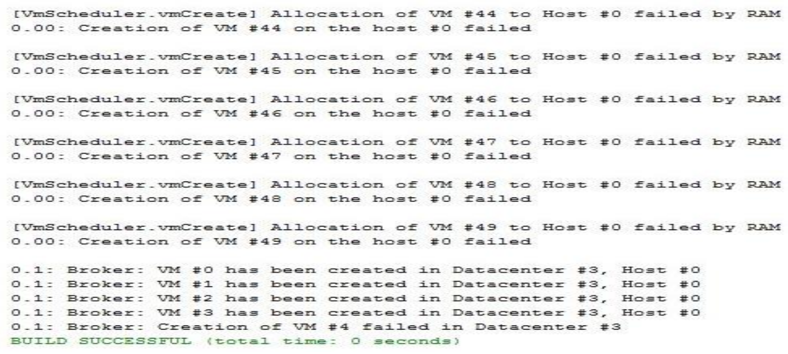


```
Experiment name: random_iqr_rs_1.5
Number of hosts: 50
Number of VMs: 25
Total simulation time: 86400.00 sec
Energy consumption: 55.88 kWh
Number of VM migrations: 5543
SLA: 0.05115%
SLA perf degradation due to migration: 0.51%
SLA time per active host: 9.94%
Overall SLA violation: 1.84%
Average SLA violation: 10.42%
Number of host shutdowns: 5083
Mean time before a host shutdown: 412.33 sec
StDev time before a host shutdown: 318.86 sec
Mean time before a VM migration: 21.17 sec
StDev time before a VM migration: 7.94 sec
Execution time - VM selection mean: 0.00006 sec
Execution time - VM selection stDev: 0.00094 sec
Execution time - host selection mean: 0.00082 sec
Execution time - host selection stDev: 0.00348 sec
Execution time - VM reallocation mean: 0.00038 sec
Execution time - VM reallocation stDev: 0.00479 sec
Execution time - total mean: 0.00848 sec
Execution time - total stDev: 0.01118 sec

BUILD SUCCESSFUL (total time: 1 minute 20 seconds)
```

Figure 23: Screenshot of IQRrs simulation with rr case2

3.3.3. IQRmc Simulation with FCFS

CASE1: No. of Hosts: 50 and No. of VMs: 50

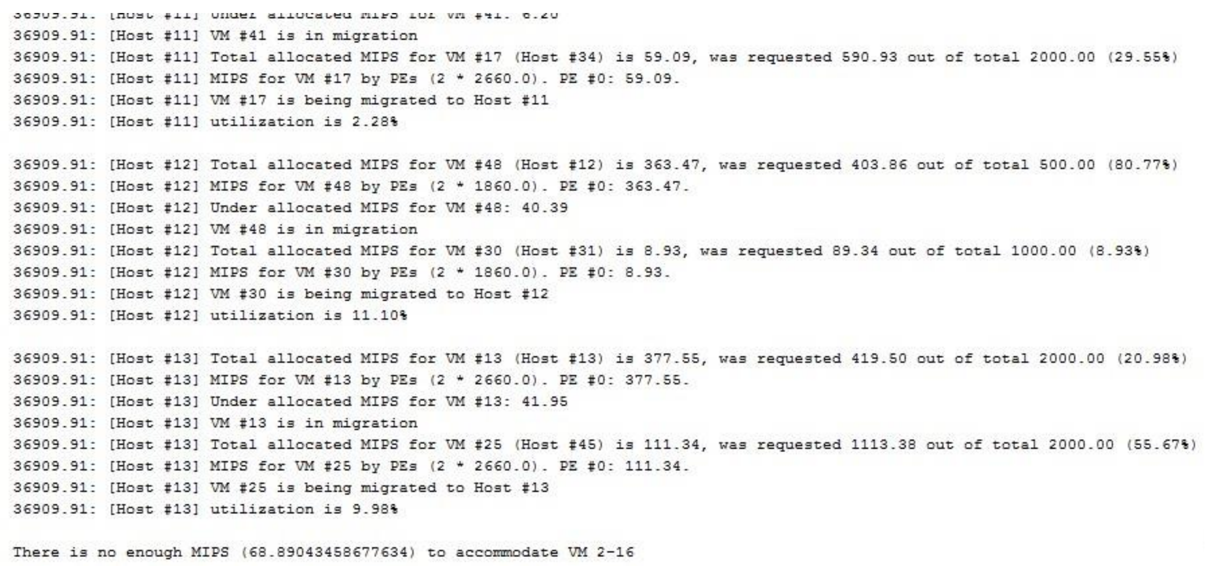


```
[VmScheduler.vmCreate] Allocation of VM #44 to Host #0 failed by RAM
0.00: Creation of VM #44 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #45 to Host #0 failed by RAM
0.00: Creation of VM #45 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #46 to Host #0 failed by RAM
0.00: Creation of VM #46 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #47 to Host #0 failed by RAM
0.00: Creation of VM #47 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #48 to Host #0 failed by RAM
0.00: Creation of VM #48 on the host #0 failed

[VmScheduler.vmCreate] Allocation of VM #49 to Host #0 failed by RAM
0.00: Creation of VM #49 on the host #0 failed

0.1: Broker: VM #0 has been created in Datacenter #3, Host #0
0.1: Broker: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker: VM #2 has been created in Datacenter #3, Host #0
0.1: Broker: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker: Creation of VM #4 failed in Datacenter #3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 24: Screenshot of IQRmc simulation with fcfs case1

3.3.2. IQRmc Simulation with RR

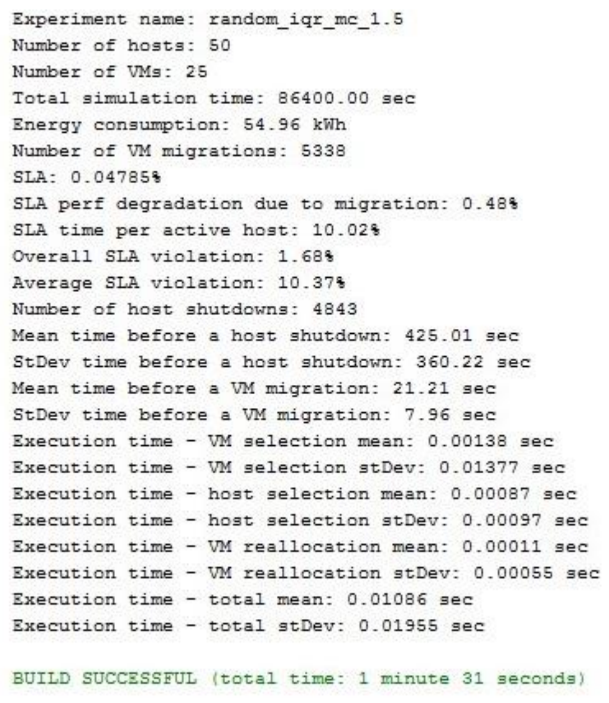
CASE1: No. of Hosts: 50 and No. of VMs: 50



```
36909.91: [Host #11] Under allocated MIPS for VM #41: 6.20
36909.91: [Host #11] VM #41 is in migration
36909.91: [Host #11] Total allocated MIPS for VM #17 (Host #34) is 59.09, was requested 590.93 out of total 2000.00 (29.55%)
36909.91: [Host #11] MIPS for VM #17 by PEs (2 * 2660.0). PE #0: 59.09.
36909.91: [Host #11] VM #17 is being migrated to Host #11
36909.91: [Host #11] utilization is 2.28%

36909.91: [Host #12] Total allocated MIPS for VM #48 (Host #12) is 363.47, was requested 403.86 out of total 500.00 (80.77%)
36909.91: [Host #12] MIPS for VM #48 by PEs (2 * 1860.0). PE #0: 363.47.
36909.91: [Host #12] Under allocated MIPS for VM #48: 40.39
36909.91: [Host #12] VM #48 is in migration
36909.91: [Host #12] Total allocated MIPS for VM #30 (Host #31) is 8.93, was requested 89.34 out of total 1000.00 (8.93%)
36909.91: [Host #12] MIPS for VM #30 by PEs (2 * 1860.0). PE #0: 8.93.
36909.91: [Host #12] VM #30 is being migrated to Host #12
36909.91: [Host #12] utilization is 11.10%

36909.91: [Host #13] Total allocated MIPS for VM #13 (Host #13) is 377.55, was requested 419.50 out of total 2000.00 (20.98%)
36909.91: [Host #13] MIPS for VM #13 by PEs (2 * 2660.0). PE #0: 377.55.
36909.91: [Host #13] Under allocated MIPS for VM #13: 41.95
36909.91: [Host #13] VM #13 is in migration
36909.91: [Host #13] Total allocated MIPS for VM #25 (Host #45) is 111.34, was requested 1113.38 out of total 2000.00 (55.67%)
36909.91: [Host #13] MIPS for VM #25 by PEs (2 * 2660.0). PE #0: 111.34.
36909.91: [Host #13] VM #25 is being migrated to Host #13
36909.91: [Host #13] utilization is 9.98%

There is no enough MIPS (68.89043458677634) to accommodate VM 2-16
```

Figure 25: Screenshot of IQRmc simulation with rr case1

CASE2: No. of Hosts: 50 and No. of VMs: 25



```
Experiment name: random_iqr_mc_1.5
Number of hosts: 50
Number of VMs: 25
Total simulation time: 86400.00 sec
Energy consumption: 54.96 kWh
Number of VM migrations: 5338
SLA: 0.04785%
SLA perf degradation due to migration: 0.48%
SLA time per active host: 10.02%
Overall SLA violation: 1.68%
Average SLA violation: 10.37%
Number of host shutdowns: 4843
Mean time before a host shutdown: 425.01 sec
StDev time before a host shutdown: 360.22 sec
Mean time before a VM migration: 21.21 sec
StDev time before a VM migration: 7.96 sec
Execution time - VM selection mean: 0.00138 sec
Execution time - VM selection stDev: 0.01377 sec
Execution time - host selection mean: 0.00087 sec
Execution time - host selection stDev: 0.00097 sec
Execution time - VM reallocation mean: 0.00011 sec
Execution time - VM reallocation stDev: 0.00055 sec
Execution time - total mean: 0.01086 sec
Execution time - total stDev: 0.01955 sec

BUILD SUCCESSFUL (total time: 1 minute 31 seconds)
```

Figure 26: Screenshot of IQRmc simulation with rr case2

3.4. Utilization After Allocation (UAA) and its Simulation

For finding host for VM to be placed in it. This is the proposed algorithm.

Take variable min_power that is assigned max value. Initially allocated host is null

```
Void utilizationAfterAllocation(vm,executedHost)
for (each host in hostList ) {
  if  (executedHosts contains host)
         continue
  if (host.isSuitableForVm){
         if (cpuUtilisation !=0 &&overutilised Host After Allocation)
         continue }
   try{
       utilization after alloaction← get Utilisation After Allocation
  if (powerafteralloaction !=-1) {
       if (powerafteralloaction < minPower ){
       minPower= powerafteralloaction
       allocatedHost=host }
catch (Exception e)
        return allocatedHost
```

CHAPTER 4

SIMULATION AND RESULT ANALYSIS

4.1 Simulation Environment

We have analyzed the performance of existing algorithm using simulations. We have designed our simulation environment in Java for which we have used Netbeans7.0 tool.

To model and schedule the different applications for cloud is a challenging task which requires different load and energy performance. To overcome this challenge University of Melbourne, Australia provided CloudSim: simulation framework that provides simulation, power to manage services and modeling the cloud infrastructure. It is designed in the JAVA and is an open source simulator and works on both windows and UNIX/Linux. The CloudSim toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited effort.

4.1.1 Advantages of CloudSim

The main advantages of using CloudSim for initial performance testing include:

(i) *time effectiveness*: it requires very less effort and time to implement Cloud-based application provisioning test environment.

(ii) *flexibility and applicability*: developers can model and test the performance of their application services in heterogeneous Cloud environments (Amazon EC2, Microsoft Azure) with little programming and deployment effort.

4.1.2 Features of CloudSim

CloudSim offers the following novel features:

(i) Support for modeling and simulation of largescale Cloud computing environments, including data centers, on a single physical computing node;

(ii) a self-contained platform for modeling Clouds, service brokers, provisioning, and allocation policies;

(iii) support for simulation of network connections among the simulated system elements;

(iv) facility for simulation of federated Cloud environment that inter-networks resources from both private and public domains, a feature critical for research studies related to Cloud-Bursts and automatic application scaling

(v) availability of a virtualization engine that aids in the creation and management of multiple, independent, and co-hosted virtualized services on a data center node

(vi) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

These compelling features of CloudSim would speed up the development of new application provisioning algorithms for Cloud computing.

4.1.3 CloudSim Architecture

Figure 16 shows the multi-layered design of the CloudSim software framework and its architectural components. Initial releases of CloudSim used SimJava as the discrete event simulation engine that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock. However in the current release, the SimJava layer has been removed in order to allow some advanced operations that are not supported by it.

The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based datacenter environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues, such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state, are handled by this layer.

The top-most layer in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities given at this layer, a Cloud application developer can perform the following activities: (i) generate a mix of workload request distributions, application configurations; (ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; and (iii) implement custom application provisioning techniques for clouds and their federation.

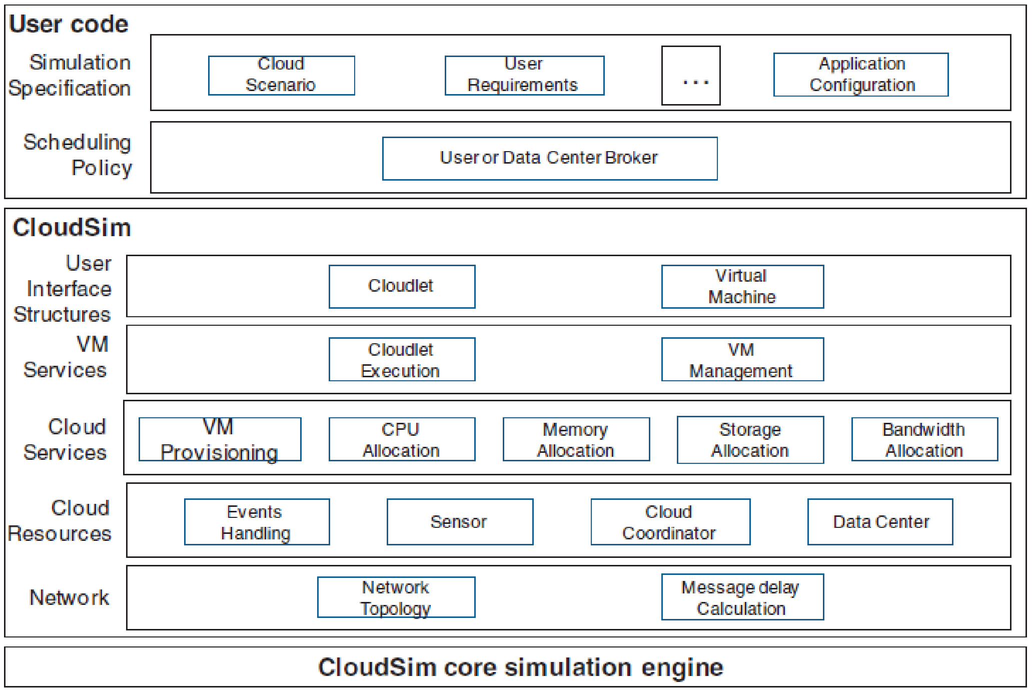


Figure 27: Layered CloudSim Architecture

The finer details related to the fundamental classes of CloudSim, which are also the building blocks of the simulator are shown in the figure given below which is the overall Class design diagram for CloudSim:

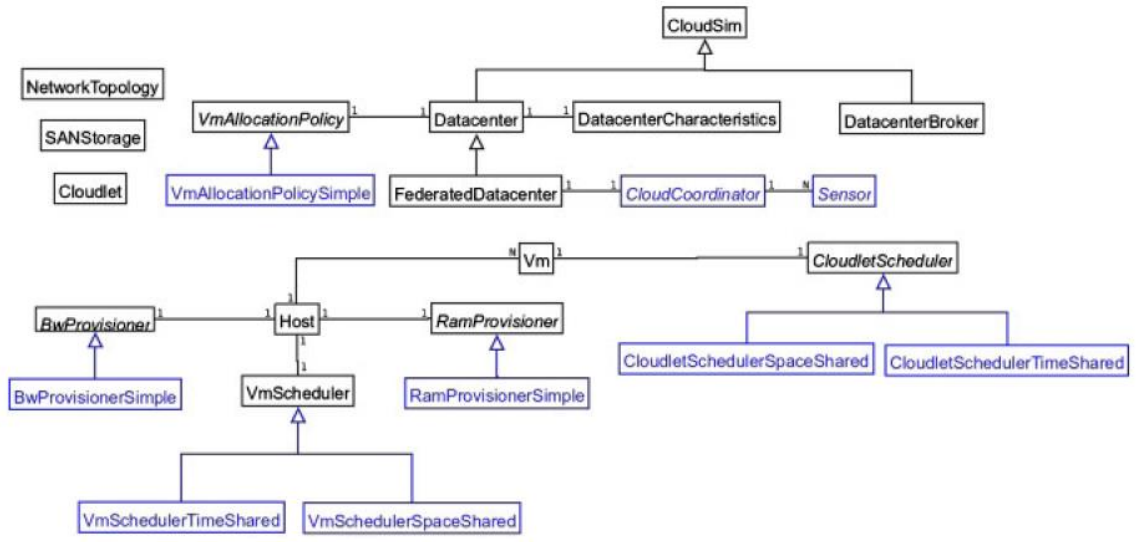


Figure 28: CloudSim Class Design Diagram

4.1.4 CloudSim core simulation framework

GridSim is one of the building blocks of CloudSim. However, GridSim uses the SimJava library as a framework for event handling and inter-entity message passing. SimJava has several limitations that impose some restrictions with regard to creation of scalable simulation environments such as:

- It does not allow resetting the simulation programmatically at run-time.
- It does not support creation of new simulation entity at run-time (once simulation has been initiated).
- Multi-threaded nature of SimJava leads to performance overhead with the increase in system size. The performance degradation is caused by the excessive context switching between threads.
- Multi-threading brings additional complexity with regard to system debugging.

4.2. Comparative Study of Existing Algorithms

Comparison charts and graphs of the studied and simulated algorithms on various parameters of quality.

Techniques	**Performance**	**Response time**	**Scalability**	**Overhead**	**Throughput**	**Resource Utilization**	**Fault Tolerance**	**Migration Time**
Honeybee Foraging	Yes	No	Yes	No	Yes	No	No	No
Biased Random Sampling	Yes	No	Yes	No	Yes	No	No	No
Active Clustering	Yes	No	Yes	No	Yes	No	No	No
ACCLB	Yes	No	Yes	No	No	Yes	Yes	No
Two-phase scheduling	Yes	No	No	No	No	Yes	No	No

ACCLB (Ant Colony and Complex Network Theory)

Two-Phase Scheduling (OLB + LBMM)

4.3. Comparative Graphs Of Simulated Algorithms with different Test Cases

	Test Case 1	Test Case 2	Test Case 3	Test Case 4
No. of VM	50	25	8	40
No. of Hosts	50	50	35	30

VM configuration:

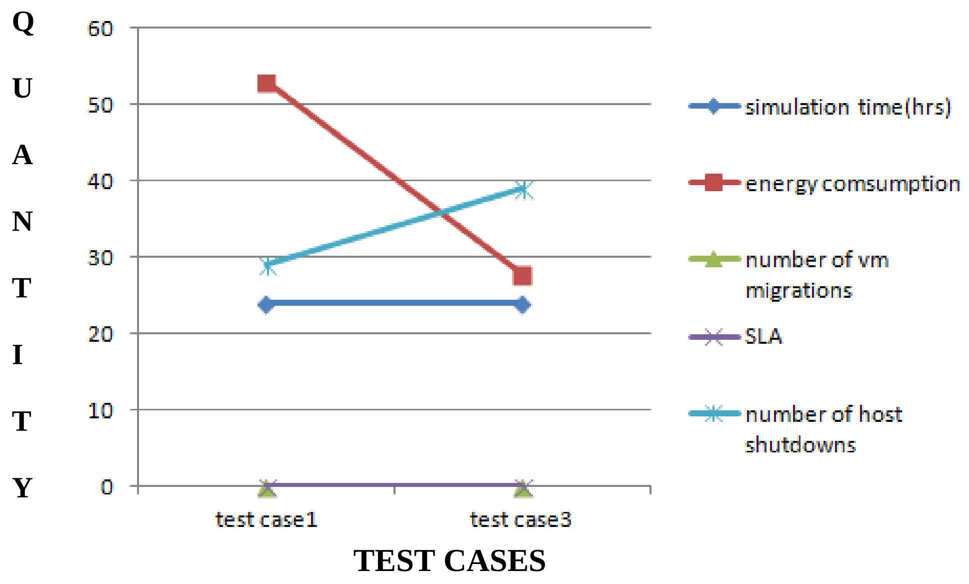
VM_MIPS {2500, 2000, 1000, 500}

VM_RAM {870, 1740, 1740, 613}

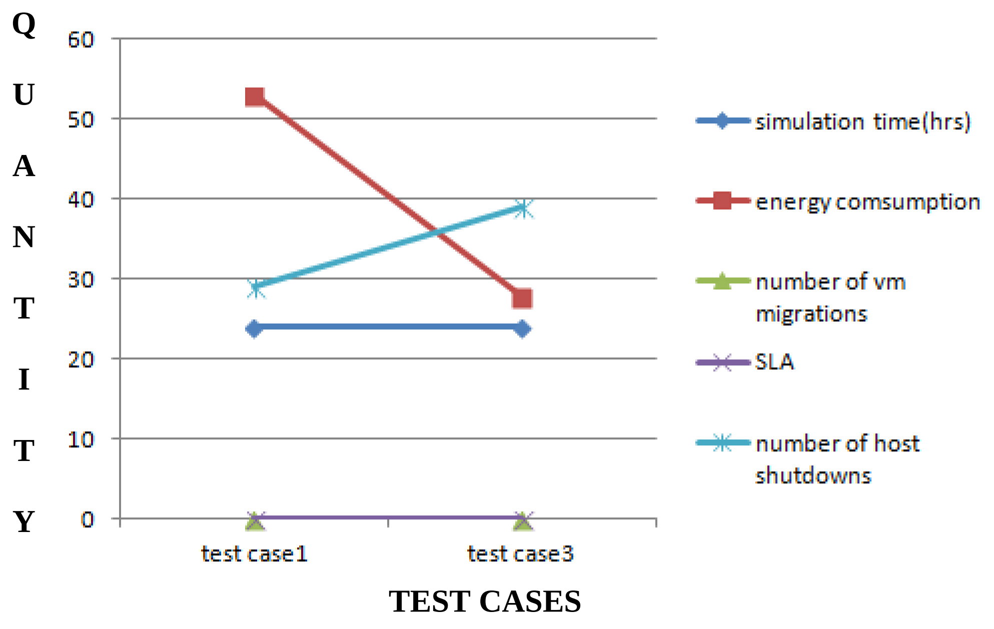
VM_BW	100000 // 100 Mbit/s
VM_SIZE	2500 // 2.5 GB

Host configuration:

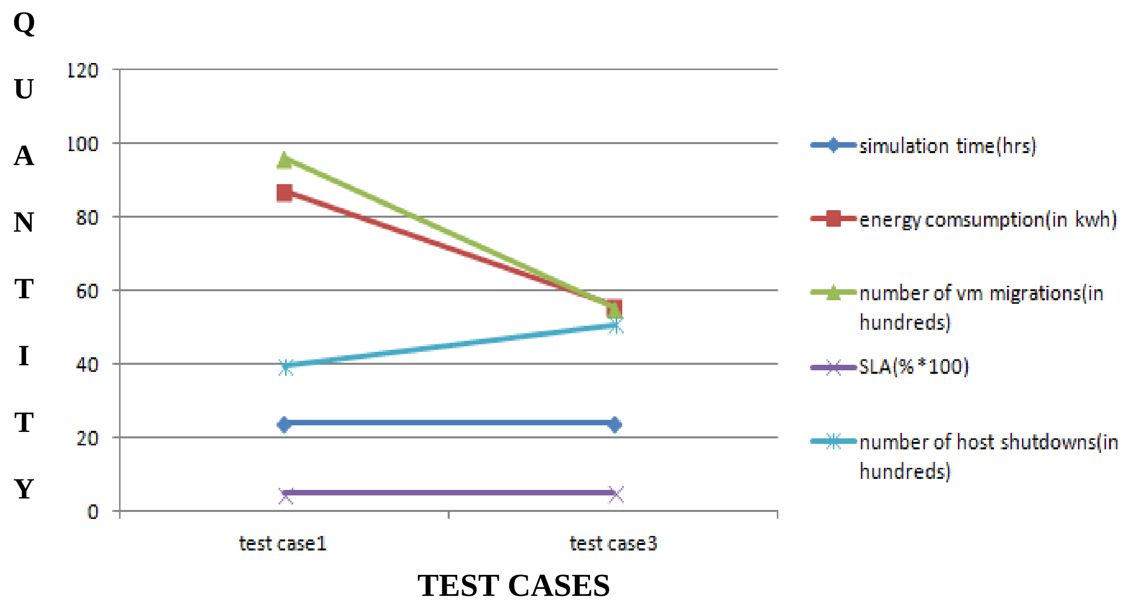
HOST_MIPS	{ 1860, 2660 }
HOST_RAM	{ 4096, 4096 }
HOST_BW	1000000; // 1 Gbit/s
HOST_STORAGE	1000000; // 1 GB



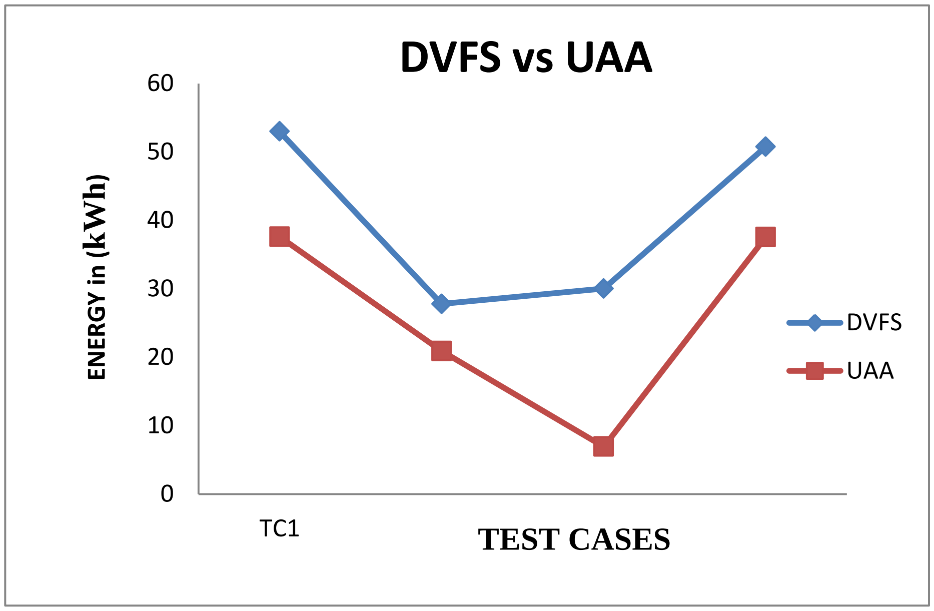
Graph1. Different test cases for DVFS simulation using FCFS In Test Case 2 creation of VM failed after 24.



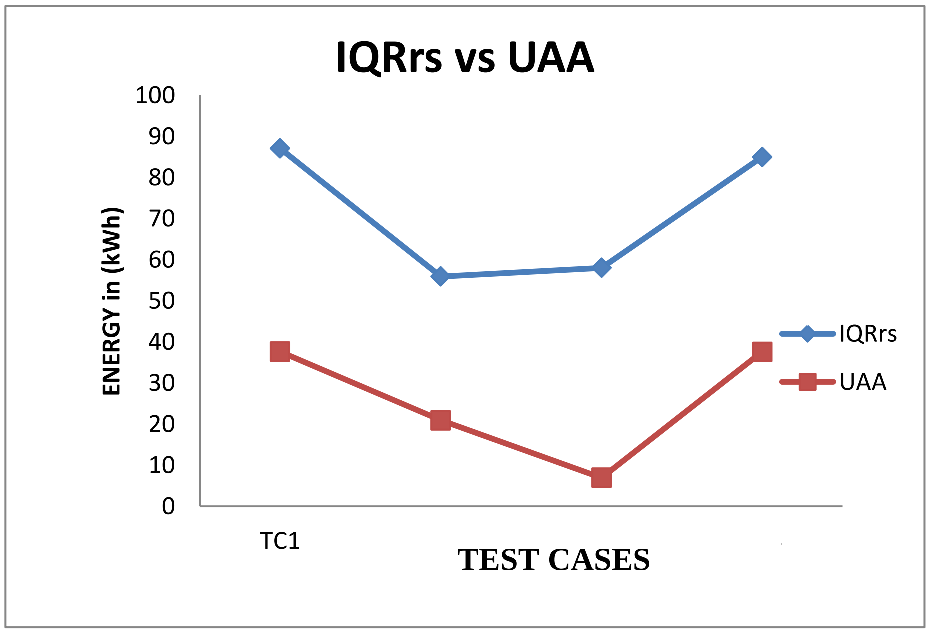
Graph 2. Different test cases for DVFS simulation using RR.



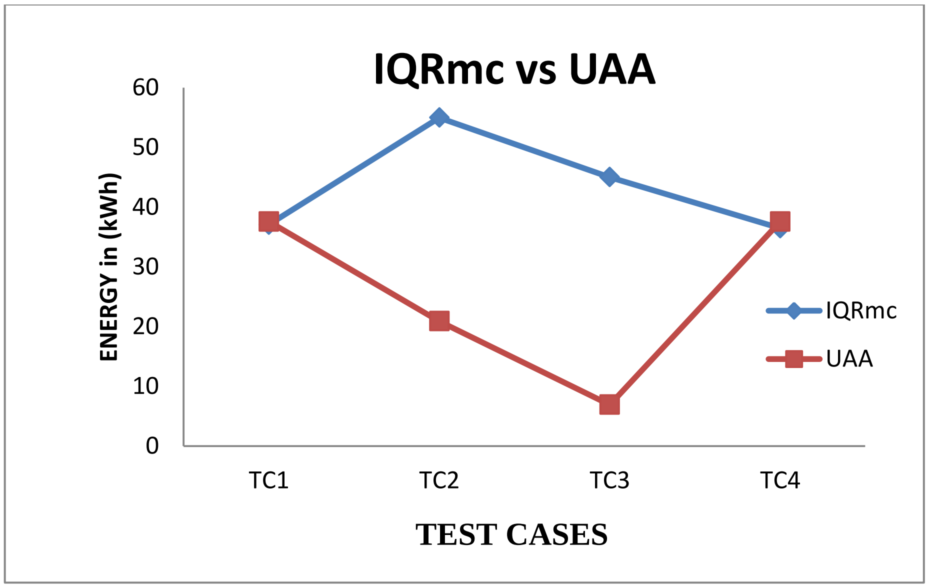
Graph 3. Different test cases for IQRrs simulation with RR.



Graph 4. Comparison of Energy of DVFS and UAA with different test cases

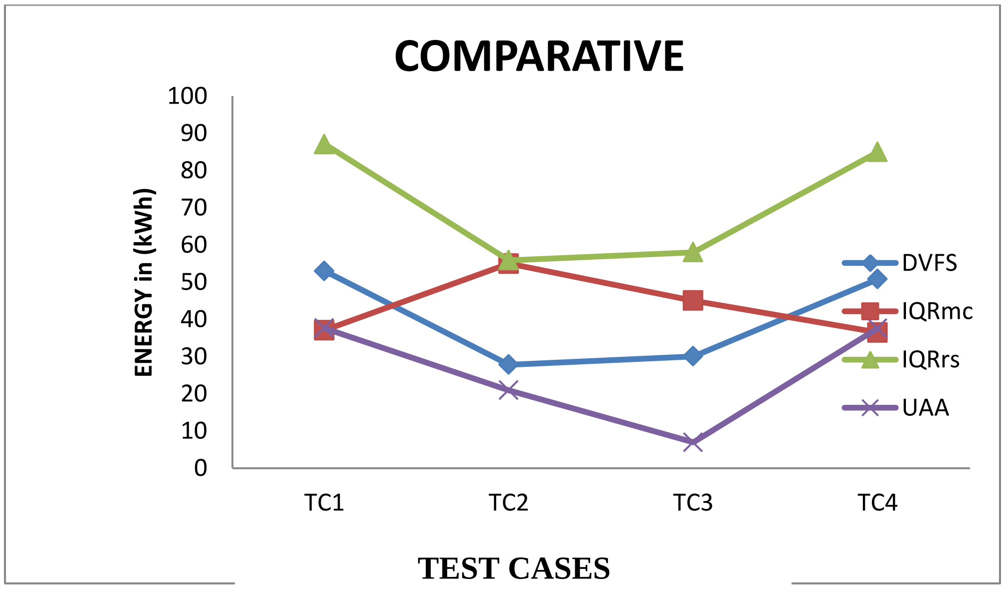


Graph 5. Comparison of IQRrs and UAA at different test cases



Graph 6. Comparison of IQRmc and UAA at different test cases

4.4. Energy comparison of all simulated Algorithms



Graph7: Comparative graph of simulated algorithms

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this report, a comparative study of the existing algorithms (DVFS and IQR) has been done using FCFS and Round Robin technique of resource allocation.. The concluding results show that there is a need for improving these existing algorithms to enhance their scalability and power efficiency parameters. The proposed algorithm names utiisation after allocation (UAA) is simulated which has resulted in enhanced energy efficiency.

5.2 Future work

In this report, proposed work presents existing techniques of handling load of servers in cloud environment. In future other parameters can be modified to achieve greater utilization and less power consumption.

REFRENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "*Above the Clouds: A Berkeley View of Cloud Computing*", EECS Department, University of California,Berkeley, Technical Report No., UCB/EECS-2009-28, pages 1-23, February 2009.

[2] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, "*Cloud Computing A Practical Ap- proach*", TATA McGRAW-HILL Edition 2010.

[3] Nikhil S. Dharane, Abhijeet S. Kulkarni, Akshay U. Bhawthankar, A.A. Deshmukh "*Load Balancing Method for Infrastructure as a Service (IaaS) In Cloud Computing: Survey*".

[4] Amandeep Kaur Sidhu, Supriya Kinger "*Analysis of Load Balancing Techniques in Cloud Computing.*

[5] Kulkarni, Gurudatt, Rani Waghmare, RajnikantPalwe, VidyaWaykule, HemantBankar, and KundlikKoli, "*Cloud storage architecture,"* In 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Bali, Oct 2012, pp 76-81.

[6] Kansal, Nidhi Jain, and Inderveer Chana, "*Existing load balancing techniques in cloud computing: a systematic review*" Journal of Information Systems and Communication, vol. 3, No.1, pp. 87-91, 2012..

[7] Randles, Martin, David Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," In IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, Perth, WA, April 2010, pp.551-556.

[8] MayankaKatyal, Atul Mishra, "*A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment*", International Journal of Distributed and Cloud Computing Volume 1 Issue 2 December 2013

[9] Luis M. Vaquero, Luis Rodero-Merino, RajkumarBuyya "*Dynamically Scaling Applications in the Cloud*", The University of Melbourne, Australia.

[10] Nikhil S. Dharane,, Abhijeet S. Kulkarni,, Akshay U. Bhawthankar,, A.A. Deshmukh "*Load Balancing Method for Infrastructure as a Service (IaaS) In Cloud Computing: Survey*".

[11] Yao, Jing, and Ju-hou He, "*Load balancing strategy of cloud computing based on artificial bee algorithm,*" In 8th International Conference on Computing Technology and Information Management (ICCTIM), Seoul, April 2012, pp 185-189..

[12] Maunika M. Ramani , Prof. Mohammed H. Bohara , "Energy Aware Load Balancing In Cloud Computing Using Virtual Machines"

[13] Anton Beloglazov, Rajkumar Buyya, "*Energy Efficient Resource Management in Virtualized Cloud Data Centers*", 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing

[14] Nidhi Jain Kansal, Inderveer Chana, "*Existing Load Balancing Techniques in Cloud Computing: A systematic Review*", Journal of Information Systems and Communication ISSN: 0976-8742, E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp- 87-91.

[15] Anton Beloglazov and Rajkumar Buyya , "*Optimal Online Deterministic Algorithm and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers*", The University of Melbourne.

[16] Li, Kun, GaochaoXu, Guangyu Zhao, Yushuang Dong, and Dan Wang, "*Cloud task scheduling based on load balancing ant colony optimization*," In Sixth Annual ChinaGrid Conference, Liaoning, Aug. 2011, pp. 3-9.

[17] Praveen Shukla, R.K. Pateriya, "IQR based Approach for Energy Efficient Dynamic VM Consolidation for Green Cloud Data Centers".

[18] Tom Guérout, Thierry Monteil, Georges Da Costa, Rajkumar Buyya, Mihai Alexandru, "*Energy-aware simulation with DVFS*", The University of Melbourne, Australia.