Design and Development of QoS Guaranteed Cloud Scheduling

A THESIS

Submitted by

REMESH BABU K R

(Reg No. 4740)

for the award of the degree of

DOCTOR OF PHILOSOPHY

DIVISION OF INFORMATION TECHNOLOGY

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY, KOCHI

APRIL 2019

CERTIFICATE

This is to certify that the thesis entitled **Design and Development of QoS Guaranteed Cloud Scheduling** submitted by **Remesh Babu K R** to the Cochin University of Science and Technology, Kochi for the award of the degree of Doctor of Philosophy is a bonafide record of research work carried out by him under my supervision and guidance at the Division of Information Technology, School of Engineering, Cochin University of Science and Technology. The content of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma. I further certify that the corrections and modifications suggested by the audience during the pre-synopsis seminar and recommended by the Doctoral Committee of Mr. Remesh Babu K R are incorporated in the thesis.

Kochi - 682022
Date: 10 – 04 – 2019

Dr. Philip Samuel
Research Guide
Professor
Department of Computer Science
CUSAT

DECLARATION

I hereby declare that the work presented in the thesis titled **Design and Development of QoS Guaranteed Cloud Scheduling** is based on the original research work carried out by me under the supervision and guidance of Dr. Philip Samuel, Professor, Department of Computer Science, for the award of the degree of Doctor of Philosophy with Cochin University of Science and Technology. I further declare that the contents of this thesis in full or in parts have not been submitted for the award of any degree, diploma, associate ship, or any other title or recognition from any other University/ Institution.

Kochi – 682022	**Remesh Babu K R**
Date: 10 – 04 – 2019	Research Scholar

Dedicated to

My Parents

Mr. B Raman

&

Mrs. K L Sarada

ACKNOWLEDGEMENTS

All praise and thanks to God Almighty for all the blessings showered on me from time to time.

First and foremost, I owe my profound sense of reverence to my supervisor and guide, **Dr. Philip Samuel**, for his patience, enthusiasm, immense knowledge and motivating nature and endless support, which paved the way for the successful completion of my doctoral dissertation. I have been extremely fortunate to have a supervisor who cared so much about my work and I cherish the opulent experience of working with him.

My special word of thanks to Dr. Sudheep Elayidom M, for being my Doctoral Committee member as well as a motivator.

I am extremely thankful to Dr. Renumol V G, Head, Division of Information Technology for providing me the facility, support and encouragement to pursue the PhD work in the department. I gratefully acknowledge all the teaching and nonteaching staff of Division of Information Technology and Division of Computer Science & Engineering, who have been very forthcoming to offer advice and help in their respective roles.

My deep felt gratitude goes to all my teachers, who introduced me to the vast expanses of knowledge, throughout my education. Without their help, I would not have been able to accomplish my professional ambitions.

I fondly remember the friendship I had with the research scholars in our Division. I am thankful to all my friends and my colleagues in Government Engineering College, Idukki. In particular, I remember with gratitude the timely support and encouragement from Mr. Ratheesh T K, Ms. Asha Ali, Ms. Mickey James, Ms. Geethu K Mohan, Dr. Preetha K G, Ms. Saritha S and Mr. Binu A. I am grateful to all my students, especially Ms. Sreelekshmi S and Ms. Arya K S, for their thoughts, wishes and prayers.

I would like to thank all the anonymous reviewers for giving critical suggestions, to improve this research.

I am deeply indebted to my parents, in-laws and my brother Rajesh Babu and sister Rekha, who have been always a source of inspiration, offering a helping hand with reassuring support in all situations that looked difficult to me.

Finally and most importantly, words are too short to express my deep sense of gratitude towards my beloved wife Ms. Krishnalekha P L and my most loving children Redhu R Krishna & Rishika R Krishna for their understanding, encouragement, patience and unwavering love. I vouch that this journey would not have been possible without their priceless and perpetual support, invaluable help and inspiration.

Remesh Babu K R

ABSTRACT

In the most generalized context, cloud computing refers to the on-demand delivery of a shared pool of virtual computing resources over a network to the remote users. These resources can be rapidly provisioned based on customer requirement. Cloud service providers try to attract more customers to increase their profit, while cloud customer expectations are good Quality of Service (QoS). The customer requirements and nature of resources are in heterogeneous nature. Scheduling is generally considered as a difficult problem of managing jobs within the given time constraint. However, the problem becomes more complicated when QoS is also considered with scheduling. QoS depends on several factors like makespan, delay, response time, over and under loaded conditions, violations in Service Level Agreement (SLA), frequent migrations, system stability and parasitic load. Cost, energy and scalability decisions are other factors that influence the performance. The objective of this thesis is to provide QoS in cloud scheduling.

We have developed a Virtual Machine (VM) placement scheme to minimize makespan. It also minimizes the storage requirement as well as power consumption. Next we have developed and tested hybrid method based on an evolutionary algorithm for VM migration through load balancing. It minimized makespan and imbalance in the cloud eco system. We developed an energy-efficient clustered load balancing for server farms for promoting green computing. It achieved energy efficiency through active physical server clustering. A novel interference aware prediction model to enhance the stability in the cloud eco system is developed and tested in real cloud. This mechanism reduced the performance interference in the cloud datacenter by predicting optimal threshold range for the maximum efficiency for the physical servers. Another contribution is the development of an SLA enforcement mechanism with auto scaling. This dynamic provisioning system with scaling policy reduced makespan, number of SLA violations, penalty cost and maximizes profit. Finally, this thesis presents an integrated SLA enforcement scheme with the aid of a prediction model. The incorporated prediction model is based on the past usage pattern and forecasts future SLA violations due to fluctuating workload. It helps in scaling decisions and resulted in reduced cost, makespan, SLA violations, and frequent migrations. All the methods mentioned above resulted in better Quality of Service in cloud scheduling.

TABLE OF CONTENTS

Chapter 4: ENHANCED LOAD BALANCING FOR VM MIGRATIONS

Chapter 5: LOAD BALANCING FOR IMPROVING ENERGY EFFICIENCY

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABC	Artificial Bee Colony Algorithm
ACO	Ant Colony Algorithm
AWS	Amazon Web Service
CIS	Cloud Information Service
CoT	Cloud of Things
CPS	Cyber Physical Systems
CSM	Cloud Scalable Multi-objective
CSP	Cloud Service Provider
DFM	Dynamic Forecast Migration
DI	Degree of Imbalance
DVFS	Dynamic Voltage and Frequency Scaling
EA	Evolutionary Algorithm
EC2	Elastic Compute Cloud
ESPP	Elastic Services Placement Problem
EWRR	Enhanced Weighted Round Robin
FFD	First Fit Decreasing
FP	Foraging Pheromone
GA	Genetic Algorithm
IaaS	Infrastructure as a Service
IABC	Interaction Artificial Bee Colony Algorithm
IoT	Internet of Things
LAMP	Linux, Apache, MySQL, PHP

MA	Memetic Algorithm
MAPE	Monitor, Analyze, Plan, Execute
MIPS	Million Instruction Per Second
MLF	Minimum Laxity First
MOP	Multi-objective Optimization
MQS	Multi Queue Scheduling
NIST	National Institute of Standards and Technology
NP	Non-deterministic Polynomial
OCCI	Open Cloud Computing Interface
PaaS	Platform as a Service
PE	Processing Element
PiA	Pareto-derived interference aware
PM	Physical Machine
PMA	Profit Maximization Algorithm
PMU	Physical Machine Utilization
PSO	Particle Swarm Optimization
PT	Processing Time
QoS	Quality of Service
RIAL	Resource Intensity Aware Load balancing
RR	Round Robin
SBP	Service-based Business Process
SA	Simulated Annealing
SD	Standard Deviation
SDN	Software Defined Network
SI	Spot Instances

SLA	Service Level Agreement
SaaS	Software as a Service
TP	Trailing Pheromone
TRACON	Task and Resource Allocation CONtrol
TTSA	Temporal Task Scheduling Algorithm
VM	Virtual Machine
VMI	Virtual Machine Image
VMM	Virtual Machine Monitor
VMU	Virtual Machine Utilization
VNM	Virtual Network Monitoring
WSLB	Weighted Signature based Load Balancing

SYMBOLS

Symbol	**Description**
θ	Pheromone evaporation rate
$\dot{P}$	Penalty
γ	Regression coefficient
λ	A cloud service
ω	Set of SLA parameters with a service λ
$\Psi(\omega)$	Number of SLA violations
R	Minimum amount of extra resources to a VM
μ	Average
σ	Standard deviation
α	Cost for SLA violation
β	Cost for Service rejection
S	Total processing power of a host

CHAPTER 1

INTRODUCTION

Contents

1.1 The Cloud Computing

In the most generalized context, cloud computing refers to the delivery of computing resources, such as compute, data resources and application softwares over a network to the remote users. One of the key attractions of cloud computing is the ability for customers to access the huge amount of computing resources on a pay-as-you go basis. According to the National Institute of Standards and Technology (NIST) [1] cloud is defined as:

"Cloud computing is a paradigm that enables on-demand network access to a shared pool of configurable virtual resources which can be rapidly provisioned and used based on the pay-per-use model".

Cloud computing allows storing data and accessing computing resources such as processing power, data, and applications over the internet instead of local computer hardware. It is a form of distributed system based on virtualization technology.

Now, cloud computing became the global computing infrastructure for business applications by providing large scale services with minimum cost [2]. The ubiquitous nature with on-demand computing facilities made it as a popular computing model. It is a promising paradigm for the computing world that offers on-demand Information Technology resources and services to the customers over the Internet. Since the customers only need to pay for the services they actually used, there is a rapid growth in the usage of cloud resources.

The cloud resources can be dynamically provisioned and reconfigured to adjust variable load (scale). The pools of resources are made available to the customers based on pay-per-use model and guarantee Quality of Service (QoS) as per customized Service Level Agreement (SLA).

1.1.1 Cloud Deployment Models

The deployment model refers to the ownership and access specification of cloud services. The cloud can be deployed using four models as shown in figure 1.1.

1. **Public cloud:** the service provider owns and operates the cloud infrastructure and services are available to the general public. Here public means any individual or a small, medium or large organization.
2. **Private cloud:** the cloud is set up for an organization solely for its own purpose. The organization owns and operates the cloud infrastructure and services are available for the employees in general and for the stakeholders of the organization who have proper access. The infrastructure may be present on-premise or off-campus.
3. **Community cloud:** a specific community may set up a cloud infrastructure for an intended purpose and shared concerns. The community may include many organizations or individuals as members. This cloud may be owned by the members of the community or maybe rented from service providers and management is performed accordingly.
4. **Hybrid cloud:** this is a combination of two or more clouds of the above categories, bound by standardized technologies for sharing and interoperations.

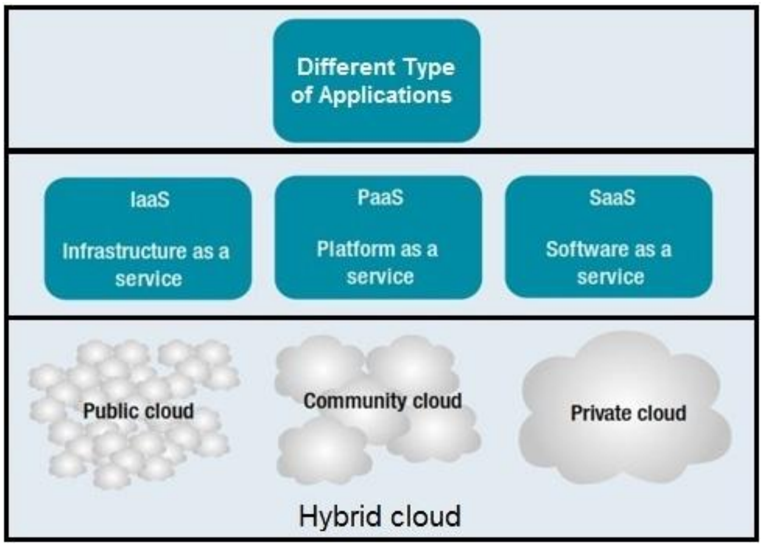


Fig 1.1 Cloud deployment models

1.2 Cloud Delivery Models

The cloud delivery model provides a specific combination of IT resources offered by a cloud provider. There are three different types of delivery models as shown in figure 1.2.

1.2.1 Software as a service (SaaS):

In this model, a complete application is offered to the customer, as a service on demand. A single instance of the service runs on the cloud and multiple end-users are serviced. On the customers' side, there is no need for upfront investment in servers or software licenses, while for the provider, the costs are lowered, since only a single application needs to be hosted and maintained.

Software or applications are provided as a service to the consumers. The software runs on the cloud environment and is accessed by consumers through well- defined interfaces such as web browsers.

The clients can be thin, and the overhead of the developing applications, hosting them, procuring infrastructure necessary for development and deployment of applications, and maintenance are eliminated for the clients. Today SaaS is offered by companies such as:

- GoogleApps by Google [4]
- SQL Azure by Microsoft [6]
- Oracle On Demand by Oracle [7]

1.2.2 Platform as a service (PaaS):

Here, a layer of software or development environment is encapsulated and offered as a service, upon which other higher levels of service can be built. The customer has the freedom to build his own applications, which run on the provider's infrastructure. To meet the manageability and scalability requirements of the applications, PaaS providers offer a predefined combination of OS and application servers, such as LAMP platform (Linux, Apache, MySQL, and PHP), restricted J2EE, Ruby, etc.

The platform necessary to develop and deploy applications and hardware are provided as services to the consumers. Consumers need not bear the overhead cost of procuring necessary platforms for their applications, getting license, updates, and renewal of licenses, etc., but have control over the configuration settings or on releasing the next version of their software.

Examples of PaaS services are:

- Force.com by salesforce.com [8]

- GoGrid CloudCenter [9]
- Google AppEngine [5]
- Windows Azure Platform [6]

1.2.3 Infrastructure as a service (IaaS):

IaaS provides basic storage and computing capabilities as standardized services over the network. Servers, storage systems, networking equipment, data centre space, etc. are pooled and made available to handle workloads. The customer would typically deploy his own software on the infrastructure.

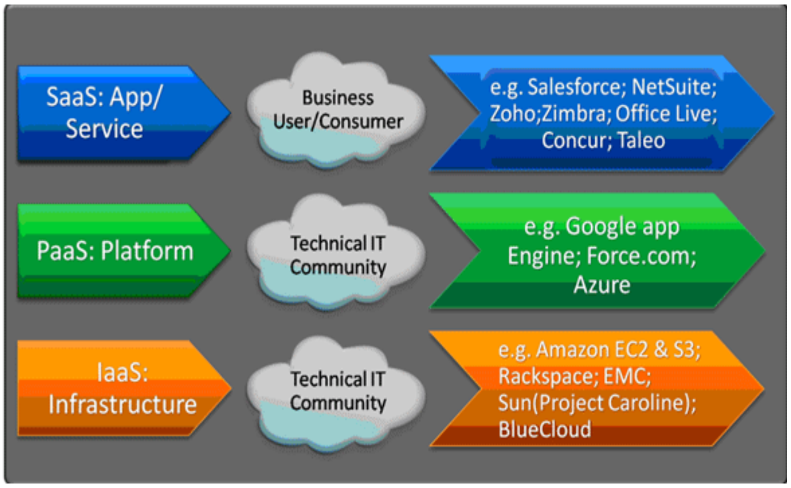

SaaS: App/ Service
Business User/Consumer
e.g. Salesforce; NetSuite; Zoho;Zimbra; Office Live; Concur; Taleo
PaaS: Platform
Technical IT Community
e.g. Google app Engine; Force.com; Azure
IaaS: Infrastructure
Technical IT Community
e.g. Amazon EC2 & S3; Rackspace; EMC; Sun(Project Caroline); BlueCloud

Fig 1.2 cloud delivery models

The resources necessary for a consumer to perform a variety of operations ranging from working with applications, developing applications, managing network of nodes, setting up networks, taking backup of data, or computers with different operating systems are provided as services. The services can be rented by individuals for personal use or by small and medium enterprises as well as multinational organizations with branches distributed across the globe.

Examples of IaaS service providers include:

- Amazon Elastic Compute Cloud (EC2) [10]
- Eucalyptus [11]
- GoGrid [9]
- FlexiScale [12]
- RackSpace Cloud [13]

1.3 Significance of scheduling

Resource management in cloud computing infrastructure is handled by Virtual Machine (VM) scheduling and it will reduce operational as well as energy cost. The scheduling is the process of allocation of different tasks to resources with high quality, considering the parameters such as makespan, energy, cost, profit, etc.

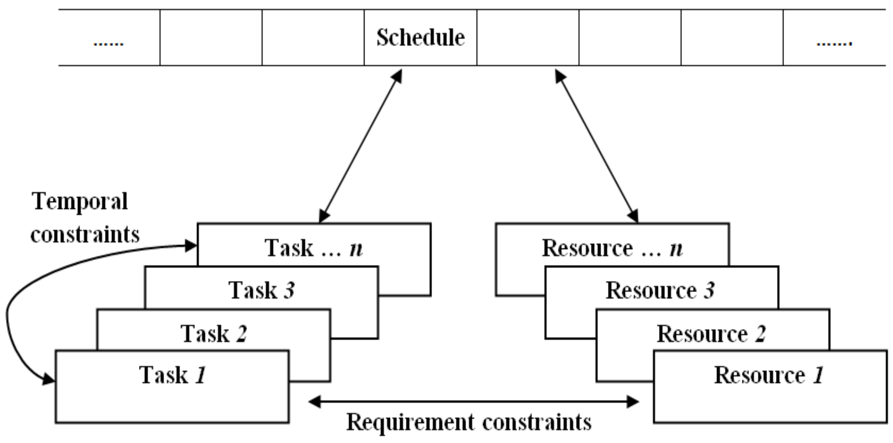


Fig. 1.3 Scheduling in Cloud.

In cloud computing, resource management is an important task in scheduling of services, customer tasks, and hardware infrastructure.

The scheduling is the allocation of user submitted tasks to particular VM provisioned in a Physical Machine (PM). When demand increases from the user’s side, then the service provider can extend their computation resources beyond their boundaries to accommodate incoming requests. Cloud needs efficient intelligent task scheduling methods for resource allocation based on workload and time. Optimal resource allocation minimizes the operational cost as well as execution time. This, in turn, reduces power and energy consumption and operational cost. Hybrid technology is needed to support customers to choose different computation offers from Cloud Service Providers (CSP). The offers from CSPs are attracted customers to promote their business and to reduce the operational cost. CSPs offer services in different categories such as subscription of services with expertise, Service Level Agreement (SLA) based, compliance, scalable and cost-effective manner.

The resource provisioning techniques decide which resources are to be made available to meet the customer requirements, while task scheduling is the process of allocating customer or user tasks to the resources based on some criteria. Resource allocation is performed by the scheduling of resources based on temporal and customer requirement constraints. In the dynamic cloud environment, both customer requirements and cloud resource status vary with time, hence scheduling based on temporal constraints is a cumbersome task. So constraints play a major role in scheduling. Proper consideration of constraints will produce a high level of QoS. Figure 1.3 gives an illustration of resource management with the scheduling of services based on constraints in the cloud.

There are several scheduling methods existing in the cloud computing, due to its multi-tenant, on-demand, elastic nature with pay-as-you-go model, but enhanced methods are necessary to improve the performance. Also, the dynamicity of cloud in resource and task scheduling gives several opportunities to the researchers. Schedulers have to consider the trade-off between functional as well as non-functional requirements to attract customers and QoS with profit.

A good resource allocation policy must avoid certain situations as follows.

- Resource contention: it occurs when more than one customer or user requests for the same service at the same time.
- Scarcity of resources: it occurs when the availability of the resource is limited.
- Resource fragmentation: if the service provider can have enough resources to accept a new request, but it is unable to allocate that request.
- Over-provisioning: The application gets surplus resources than the demanded one.
- Under-provisioning: The application is assigned with less number of resources than demanded.

1.3.1 Cloud Properties that Affect Scheduling

Certain factors that affect cloud scheduling depends upon the nature of cloud resources. These factors are homogeneity and heterogeneity of cloud resources. The elastic nature of cloud resources is also an import factor. Scalability of resources auto scaling properties is also crucial in the scheduling process.

1.3.1.1 Homogeneity

In a homogeneous cloud, the entire software stack including the hypervisor, intermediate cloud stack, and customer portal are from the same service provider. So here management is simple since the entire things are from a single provider. Since everything comes in a pre-integrated manner, if anything goes wrong, just one party holds the responsibility. When one CSP is in the possession of so much power, customers become dependent on the same provider’s technical and commercial strategy. The advantage of this kind of cloud environment is that customers can able to specialize in a CSP’s tool. While administrators can easily cover for each other within this strategy, the downsides are different. The features are available on the technical side, but which is exclusively developed by the particular service provider. Besides, when a customer or user is “locked-in” to one service vendor strategy, resources can be easily delegated despite changes in the pricing structure. This belongs to the commercial side advantage.

1.3.1.2 Heterogeneity

To increase performance and attract more customers, CSPs are adding different types of computing resources with increased memory and storage capacities. Thus heterogeneity improves the overall cloud performance and its power efficiency. Customers are often looking for sophisticated high-end infrastructure such as high speed processors, with low cost. The moves towards green computing standards are now focusing on energy consumption. So public CSPs are now implementing different mixtures of architecture for their infrastructure to improve power efficiency. This complex heterogeneous cloud data centre needs more powerful dynamic algorithms for resource and task management. Internets of Things (IoT) implementations are now rapidly increasing around the world. These IoT devices generate a massive amount of data and need more processing power to analyze it. Hence heterogeneous cloud implementations are necessary for the successful IoT and related Cyber Physical Systems (CPS) implementations.

1.3.1.3 Elasticity

In cloud computing, elasticity is defined as the degree to which a system is able to adapt workload changes by provisioning and de-provisioning resources in an automatic manner such that, at each point in time the available resources match the current demand as closely as possible. Elastic cloud infrastructure provides a cloud computing environment with greater flexibility and scalability. Amazon Web Service (AWS) facilitates web service scalability.

Elasticity is the ability to fit the resources needed to cope with workloads dynamically usually in relation to scale out. When the load increases, adding more resources by scaling and when demand wanes, the system shrinks backs and removes unused resources. Elasticity is mostly important in cloud environments where pay-per-use and don't want to pay for resources that customer does not currently need on the one hand, and want to meet rising demand when needed on the other hand. Elasticity adapts to both the "workload increase" as well as "workload decrease" by "provisioning and de-provisioning" resources in an "autonomic" manner. Intelligent algorithms that detect workload necessities will aid in this situation.

1.3.1.4 Scalability and Auto Scaling

Scalability is the ability of the cloud ecosystem to accommodate larger workloads by adding more resources either making hardware stronger (scale-up) or adding additional nodes. Scalability is performed before the increase in workload by adding additional resources or to perform well before to meet the required QoS. This enables a CSP to meet expected quality demands from the customers or to meet SLA requirements for services with long-term, strategic needs. Auto scaling mitigates the resource contention and delay in processing customer or user tasks. It aids CSPs to offer a high level of services on-demand with customer satisfaction. By scaling-out instances seamlessly and automatically when demand increases, better resource management can be done. By turning off unnecessary cloud instances automatically, CSPs can save money when demand reduces thereby achieves energy consumption. Also, it can replace unhealthy or unreachable instances to maintain higher availability for customer applications.

Auto scaling helps to ensure the availability of the right quantity of computing resources to handle customer requirements, by adding or removing resources depending on the usage. It is one of the properties of cloud computing to measure the quality of service (QoS) and performance. The capacity of the resource is scaled up and scaled down during the demand-supply of customers. Auto scaling helps to reduce the cost of computation according to resource usage and can provide a high level of services with customer satisfaction.

During the scale-out process, VM instances are provided seamlessly and automatically while during the scale-in process the unneeded instances are turn-off automatically when demand decreases thus save energy and money. Another advantage is that it replaces unhealthy or unreachable instances to maintain higher availability of customer applications. Thus on-demand cost-effective computing with seamless execution is possible in the cloud.

Figure 1.4 shows the auto scaling by configuring resources either allocate instances to new VMs or schedule to the existing computational resources.

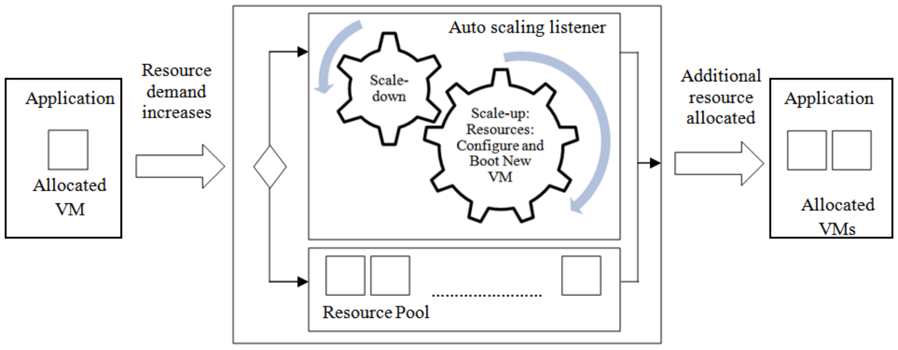

Application
Allocated VM
Resource demand increases
Auto scaling listener
Scale-down
Scale-up: Resources: Configure and Boot New VM
Resource Pool
Additional resource allocated
Application
Allocated VMs

Fig. 1.4 Auto scaling in a cloud infrastructure

1.3.2 Scheduling Constraints

Even though the cloud offers low-cost computing facilities, the customer concern while adopting cloud as their computing platform is cost, time and other QoS parameters. The service providers always concern about their profit and energy consumption. Here we are interested in performance oriented cloud scheduling that enables a specific performance targets with minimized resource consumption.

1.4 Service Level Agreements

The QoS requirement formally described in terms of an SLA specification [14]. In order to provide customer requested QoS, Infrastructure as a Service (IaaS) providers plays a major role. To maintain better performance and prevent breaches in SLAs, the IaaS providers must focus on virtualization, the fundamental building block of Cloud infrastructure.

Usually, a Cloud SLA spans over many jurisdictions, with different legal applications, especially the personal data hosted in the data center. Also, there is a need for different SLA terminology and models for different type of service providers. So it is difficult to maintain a common format for SLA for comparison. In our study, we have considered the following parameters for SLA statements covers time including deadline requirements, cost and penalty, memory requirements, storage requirements and network parameters like delay.

1.5 QoS Oriented Cloud Scheduling

As with any service, such as household utilities, QoS plays a critical role in ensuring that a customer or an end-user receives the service for which they have paid [3]. QoS for this research is defined as resource control mechanisms that guarantee a certain level of performance and availability in terms of makespan including deadline requirements, maintaining SLA, stability, cost of computation, etc.

Scheduling is generally considered as a difficult problem of managing jobs within the given time constraint. However, the problem becomes more complicated when QoS is also considered with scheduling. QoS depends on several factors like makespan, delay, response time, over and under loaded conditions, violations in Service Level Agreement (SLA), frequent migrations, system stability, and parasitic load. Cost, energy and scalability decisions are other factors that influence the performance.

There are a number of challenges facing to assure QoS in clouds. The two core challenges involve first, the guarantee of resource reservation by a binding agreement and second, the continued provisioning of a resource to specified requirements. In the context of Clouds, this translates to challenges in service provider interoperability where unification of resource control mechanisms and the resource types provisioned require standardization and additionally in challenges a service provider must face with regards to managing their resources efficiently and in selecting an appropriate software stack to meet QoS requirements pertaining to the performance and availability of provided resources.

1.5.1 Quality factors

In cloud QoS oriented scheduling depends on time, financial, SLA, stability and scalability factors.

1.5.1.1 Makespan

In cloud, most of the applications are deadline constrained, so it has to complete within the stipulated time. Customers submitting tasks with deadline constraints are mainly considered makespan or completion time as the quality parameter. All the time-dependent parameters such as response time and execution time are important factors in achieving better QoS.

1.5.1.2 Financial

Customers always prefer high-end computing facilities at a low cost. The financial constraints are applicable to both customers and providers. Customer always seeks for low cost with quality while providers trying to increase their business by attracting more customers so as to maximize their resource utilization and profit. If a service provider is able to provide high-end computing resources to their customers within their economic limit, it is a positive thing in achieving good QoS.

1.5.1.3 Service Level Agreement

The purpose of SLA is to assure the QoS to the customers. The CSPs that offer services to the customers by maintaining assured QoS in the SLA. Any violations in the agreed conditions will degrade the performance of the provider. So minimizing or avoiding SLA breaches is another QoS factor.

1.5.1.4 Stability

The performance stability can be achieved through a good load balancing mechanism. The performance drops off due to frequent load balancing in the cloud data center. i.e transfer of computation from one location to another or context switching affect or cause a delay in completing assigned tasks. So the scheduling mechanisms should consider the impact of performance fluctuations and mitigate it with efficient load balancing mechanisms.

1.4.1.5 Scalability

In cloud, scalability is the ability of service provider to expand their infrastructure to handle the increased workload. With an intelligent auto scaling mechanism, timely scaling of resources can be done to avoid SLA breaches.

In general, to attract more customers, CSPs attempt to provide more sophisticated services with QoS. For ensuring QoS, CSPs need more accurate resource management services to process customer submitted tasks. E.g. Amazon’s Elastic Compute Cloud (EC2), provides an opportunity to auction based spot pricing. So the techniques to handle spot prices will increase the quality of the scheduling process.

1.6 Motivation

Most of the cloud scheduling techniques proposed so far is based on time and cost parameters [19, 21, 23, 24, 26, 27, 28, 29]. Other parameters such as agreed conditions in the SLA, load balancing, VM migrations and energy considerations are also important factors that affect the scheduling process.

The cloud computing has presented new opportunities to the customers and application developers. They can benefit from the cloud computing paradigm in-terms of economies of scales, commoditization of assets and conformance to programming standards. Its advantages such as low cost in pay-as-you-use criteria, scalability, and elasticity quickly attracted several business organizations.

The utility type of delivery of services and instant pricing methods termed it as a business model for computing services. So, economic consideration is the primary issue in this model. Service providers always look for profit and maximum utilization of their resources with minimization of operational cost, energy, while consumers focus on better quality oriented service with minimum cost and time. It is quite easy when the cost is considered as the primary factor for scheduling [32, 35, 37], but other factors are more important in maintaining the quality of service.

The dynamicity of cloud makes resource management and task scheduling as a cumbersome task. There are several scheduling methods existing in cloud computing, due to its multi-tenant, on-demand, elastic nature with pay-as-you-go model, but these methods pose several challenges in the area of Quality of Service (QoS) management. Since QoS is the fundamental right for cloud customers, who expect service providers to deliver the announced or agreed qualities, the cloud providers should find the right tradeoffs between QoS levels and operational costs. So, more sophisticated methods are required to improve the QoS scheduling. Proper scheduling reduces the operational cost and response time in the cloud.

Schedulers have to consider the trade-off between functional as well as non-functional requirements to attract customers and QoS with profit. In the large scale distributed systems like cloud, the efficiency of scheduling algorithms is crucial for better efficiency and resource utilization. The performance of the current state-of-art algorithms needs improvement to address this issue. So workload maximization mechanisms are needed to increase the profit of service providers.

When demand for the services and users change in real-time, there is a need for dynamic resource provisioning methods. The challenges to resource provisioning include the distributed nature of resources, uncertainty, and heterogeneity of resources. Few articles addressed the load balancing method to improve the performance [58 - 61]. Due to dynamic nature, resource capacity aware methods try to reallocate customer requests to better physical servers to improve performance. These frequent reallocations cause some delay to restart the processing at new locations. Ultimately this causes performance degradation in the makespan and thereby decreases in overall performance.

The VM placement and live migration are trendy method to balance the load which is achieved by different heuristic and hybrid algorithms and optimization techniques. Frequent migrations are still a problem to be resolved. The reallocation can be done by load balancing techniques to get optimal results. Thus, there is a necessity of better load balancing techniques in the cloud.

Green computing is the latest buzz word in the computing industry. Data centers need huge power to run their infrastructure and associated cooling facilities. In order to cool down the temperature due to the operation of large server farms, proper air cooling and circulation equipment are installed in data centers. Server consolidation techniques will reduce the number of servers in the active state, so that power consumption for servers and related cooling equipment can be reduced. Too much workload on a server will result in the degradation of makespan and response time. i.e., adopting green computing and increasing resource utilization should not degrade the quality of service delivered or cause any violations in the agreed conditions in the SLA. So there is a need to improve the scheduling process by considering the tradeoff between energy and service quality. In particular sophisticated scheduling mechanism is needed to address this issue.

Simultaneous optimizations of all parameters are difficult due to the contradictory effect of each one. E.g., time and cost can't be achieved together. When we try to reduce computation time, it needs powerful servers to complete the task and these powerful machines cost more than slower servers. Using the multi-objective optimization method this type of situation can be studied to obtain a better solution.

It is also a fact that for further enhancement in this field, some challenging issues like performance interference are to be focused. Energy optimization, promotional offers from providers such as spot instance price, QoS and SLA considerations are major concerns that need more attention and improvement for scheduling in cloud data centers.

Guaranteeing SLA is the key task of a good scheduling mechanism in maintaining QoS requirements. A proper SLA ensuring mechanism is needed to ensure whether the provider delivers as in the agreement. In order to ensure SLA, an SLA violation monitor mechanism with penalty enforcement is needed. Applying penalty for each SLA breaches will be a strong way to guarantee SLA conditions. A good scheduling scheme is essential to address SLA management.

Auto scaling of resources in cloud computing allows resource provisioning dynamically and improves performance. The scalability of the cloud increases the chances to allocate more users and minimize SLA violations. Scalability helps to maintain QoS when the demand of services varies with real-time computational environment. The energy, delay, deadline, time and cost affect the scalability and these issues are to be addressed in detail for load balancing and VM placement.

In nutshell, the following are the issues in the existing cloud scheduling:

Inefficient makespan handling procedures that cause delayed completion of customer requests.

Inadequate load balancing for virtual machine migration methods results in long makespan and a large number of migrations.

Inefficient energy consumption methods increase electricity usage and operational cost.

Lack of methods to ensure system stability caused due to frequent VM migrations that reduce QoS delivered.

Lack of auto scaling mechanisms with SLA enforcement which results in pure QoS.

Lack of integrated methods to handle makespan, migrations with stability, SLA with auto scaling and reduced cost.

1.7 Problem Statement

To design and develop cloud scheduling techniques that guarantee Quality of Service.

1.8 Research Objective

Creating scheduling algorithms that confine the customer's practical needs and constraints would be extremely useful in the distributed cloud systems. A scheduling policy which will be beneficial to both service provider, as well as customers is needed. As a part of this work, we have designed and implemented policies that will improve the scheduling performance considering makespan, cost, energy, stability, SLA and other Quality of Service (QoS) requirements.

In order to ensure the quality of service delivered in the cloud, the following objectives are addressed in this thesis.

- To develop a method to handle makespan.
- To develop an efficient load balancing policy for handling VM migrations.
- To develop a cluster-based load balancing for improving energy efficiency.
- To enhance the stability of the cloud ecosystem with interference prediction.
- To develop a scheduling method to enforce SLA with auto scaling.
- To develop an integrated SLA enforcement method with reduced cost.

1.9 Thesis Organization

This thesis deals with the problems in makespan, load balancing, energy, service level conditions and auto scaling in cloud computing. We have organized this thesis into nine chapters. The rest of the thesis is organized as follows:

Chapter 2 - presents a survey of different scheduling, load balancing, and resource provisioning methods in Clouds. In this chapter, a detailed classification and a correlation taking into account different criteria of the overviewed literature/methods are exhibited and issues in each method are tabulated.

Chapter 3 – proposes a Virtual Machine placement mechanism for handling makespan. This method is based on the principle of Bin packing method. It uses a Best-fit – Remaining-fit approach for VM placement in the datacenter.

Chapter 4 – proposes an enhanced bee colony based algorithm for scheduling and load balancing, to handle VM migrations.

Chapter 5 – proposes an energy-aware clustered load balancing algorithm. In this, an energy-aware clustered load balancing system in which, heterogeneous cloud resources are grouped into different clusters, by using a partitioning based clustering algorithm.

Chapter 6 – proposes an interference aware prediction mechanism in the cloud. Here the proposed model also predicts the optimal load and threshold range for each physical server using the Pareto principle.

Chapter 7 – proposes a Petri Net based scheduling model in an elastic cloud to enforce service level agreements. Using the property of Petri Nets, a model is developed to aid auto scaling process.

Chapter 8 – this chapter deals with an integrated approach for SLA aware scheduling and load balancing method. It covers a prediction model based on the past usage pattern and that aims to provide optimal resource management without the violations of the agreed service level conditions in cloud datacenters

Chapter 9 - concludes the thesis with a summary of the contributions and discussion on future research directions.

CHAPTER 2
LITERATURE SURVEY

Contents

2.1 Introduction

The aim of a cloud scheduling model is the optimal allocation of resources to the tasks. The optimal allocation is to ensure the conditions mentioned in the service level agreements to deliver better quality of service. Generally, scheduling algorithms are classified into static and dynamic methods. We have considered and reviewed scheduling models based on parameters, VM placement, load balancing and dynamic-adaptive methods as shown in figure 2.1. It can be also classified based on the optimization method used. This classification is shown in figure 2.2 and a detailed description is given in section 2.6.

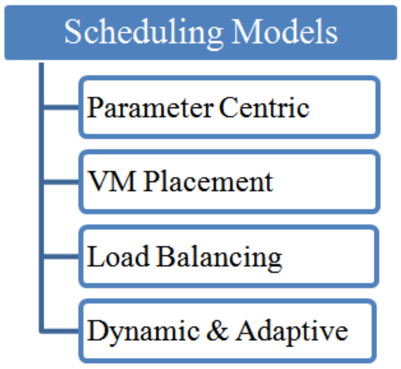


Fig. 2.1 General classification of scheduling models.

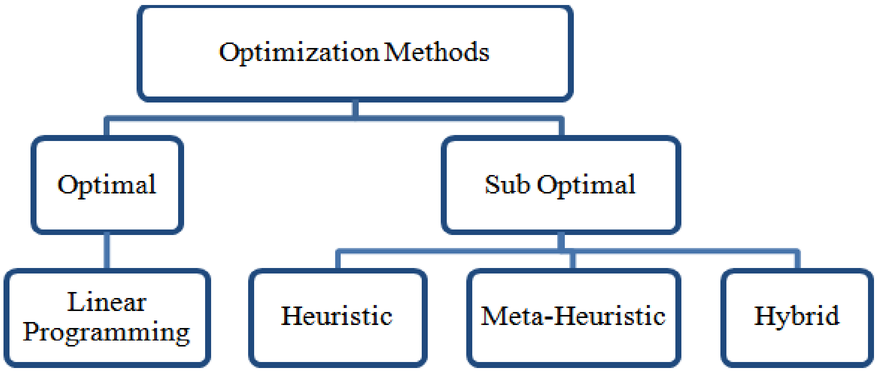


Fig. 2.2 Scheduling models based on optimization methods.

2.2 Parameter Centric Methods

The primary function of a cloud resource scheduling mechanism is to identify the suitable resources for scheduling the apt workloads on time and to increase the efficacy of resource utilization. An optimal resource-workload mapping is required for the efficient performance of scheduling methods. The main aim of cloud scheduling algorithms is to achieve some user-specified parameters such as low makespan, deadline achievement, low cost, increased system stability, etc. At the same time, these methods have to improve the overall performance of the cloud. Some methods are based on the conditions in the service level agreement. Both customers' and providers' requirements are to be considered for efficient resource allocation. The service providers always looking to increase their profit and reduction in operational cost, mainly power or energy consumption, while the consumers focused at cost and good quality of service and experience. Our literature review emphasizes resource scheduling algorithms based on different scheduling parameter centric objectives or criteria. These parameter centric scheduling objectives are shown in figure 2.3.

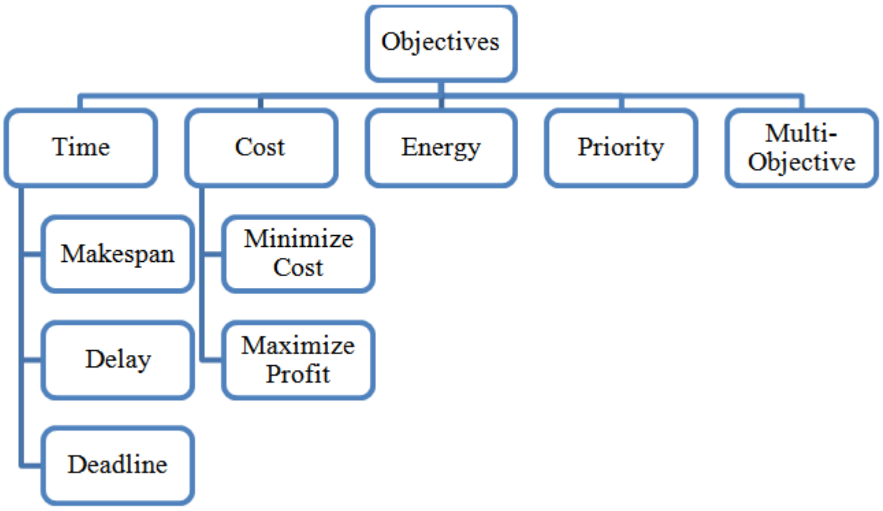


Fig. 2.3 Parameter centric scheduling objectives

Makespan: It is the total completion time taken to complete a user-submitted task. Most of the algorithms mentioned in this survey are focused on makespan as an important parameter.

Delay: It is one of the important factors in measuring the quality of service. Delay in giving responses to the customers is one of the parameters considered in this review.

Deadline: Usually the scientific workflows submitted to the cloud are to be completed within a specific time. This survey considered a sufficient number of deadline constrained papers for the comparison.

Cost: The main objective of the cloud is to minimize the cost of computation. The algorithms try to minimize the usage cost or try to provide more efficient service to the customers with the amount they spend to hire the service.

Profit: While offering low-cost services to the customers, CSPs are trying to maximize their revenue by attracting more customers. This is usually done by giving different offerings to the customers and maximizes their resource utilization rate.

Energy: Consumption of energy is crucial in reducing operational costs. One of the main costs incurring in running a cloud datacenter is energy cost. Most of the recently proposed methods are given keen attention to power utilization and energy consumption.

Priority: Since different types of customers need a vivid variety of services with varying preferences, priority is an important factor in resource scheduling.

Multi-Objective: The recent advancements in cloud scheduling methods have given attention to multiple criterions in task scheduling. These criterions are sometimes contradictory, so a trade-off is needed between different solutions produced by the scheduler.

2.2.1 Makespan

Makespan or completion time is the total elapsed time is the difference between the time of submission of a task to the provider and its completion. Usually, it is the sum of execution time, delay in communications, response time, migration time, etc. Scheduling focus on to reduce completion time [15] and to increase the maximum utilization of resources [16, 17]. We have analyzed several makespan oriented scheduling mechanisms and the details are summarized in table 2.1. These papers failed to address migration problems. Migrations cause complex interactions between different entities in the cloud, which creates delay and finally it adversely affects the overall performance of the system.

Table 2.1: Makespan

Paper	Method	Parameters	Highlights	Limitations	Environment
[15]	Fully Polynomial Time Approximation Algorithm (FPTA)	Migration time Makespan Transmission rate Bandwidth	Load balancing Low transmission rate	High SLA violations	Simulation
[16]	VM migration algorithm	Cost Migration time Resource utilization time	Load balancing Maximize resource utilization Minimum service interruption	Inefficient	Simulation
[17]	Cloud based Workflow Scheduling (CWSA)	Makespan Cost	Minimum completion time (MCT)	Service interruptions	Simulation
[18]	Map reduce framework scheduling in Hadoop	Makespan Workload	Dynamic slot configuration feedback Control-based workload estimation	Sub optimal solutions No load balancing	Real

2.2.2 Delay

A good scheduler should consider the delay in processing of user-submitted tasks and the depreciation while evaluating the CSP services. Queuing delay analysis [19] is one such method that accounts for both delay-sensitive and delay-tolerant applications. They used an optimal pricing strategy, based on profit maximization problem, which is non-convex in nature. The methods proposed for multi-cloud in [20, 21] causes additional delay and cost occurs due to inter-cloud communications. Its' performance improvement and financial savings are still significant than single cloud systems.

Table 2.2: Delay

Paper	Method	Parameters	Highlights	Limitations	Environment
[19]	Pricing algorithm	Profit Delay Cost	Delay tolerance	High energy consumption No SLA	Simulation
[20]	Resource allocation algorithm	Delay Cost	SLA constraints Multi-cloud resource allocation	No priorities No load balancing	Simulation
[21]	Profit maximization	Delay Cost Profit	Profit maximization Delay bound	Service interruption	Simulation
[22]	VM scheduling algorithm	Delay Buffer size Power	Minimum delay Minimum power consumption High QoS	No load balancing Homogeneo us resources	Real
[23]	Computation offloading with energy constraints	Delay Communication cost Computation cost Energy	Delay tolerance Minimum energy consumption	Frequent service interruption Unreliable	Simulation

Cloud scheduling is a cumbersome task due to the uncertainty in the arrival of tasks with guaranteeing service [20]. The profit maximization problem can be solved by a Profit Maximization Algorithm (PMA) and it provides a temporal task scheduling, which can dynamically schedule all the arrived tasks that can be in private or public clouds [21]. Most of the existing scheduling algorithms are pre-emptive in nature and it causes frequent context switching [10, 21]. This is due to context switching need a certain amount of time and energy for saving and loading the registers and mapping of respective memory, updating various tables and lists, etc. This again is a cause for an increase in energy usage, delay, and CPU overhead. The method proposed in [23] tried to mitigate these problems but performance is poor. Works related to delay aware methods are tabulated in table 2.2. In a nutshell, these methods tried to reduce the delay, but the overall performance is very low.

2.2.3 Deadline

In maintaining QoS, the deadline of a task is a crucial parameter. If the applications are deadline constrained, meeting its' time limit is critical and it is also a fact that there is no incentive if the application finishes the task earlier. Meeting an application's deadline requirement with the least number of resources will increase customer satisfaction as well as providers' revenue [24].

Table 2.3: Deadline

Paper	**Method**	**Parameter**	**Highlights**	**Limitations**	**Environment**
[24]	Minimal Slack Time and Minimum Distance (MSMD) algorithm	Execution time Cost	Minimize makespan Instance hour minimization Auto-scaling	Low efficiency	Simulation
[25]	Min-Min algorithm Heuristic algorithm	Execution time Cost Deadline	Optimized parameter-based sweep workflow	High execution time	Simulation
[26]	Heuristic algorithm Minimum Average Cost First (MACF)	Time Cost	Time slot filtering Greedy and fair-based scheduling	Pricing interval not considered No load balancing	Simulation

Papers [25] and [26] proposed an intelligent mechanism to meet the deadline constraints. In method [25] the workflow was executed in multiple VM instances. They have evaluated the system with different task mapping heuristics. Their experiment results show that the proposed technique is able to lower cloud usage cost when the time constraint is relaxed but have low efficiency. Also, these methods don't use load balancing mechanisms. We have tabulated the findings in table 2.3.

2.2.4 Cost and Profit

Since cloud deals with diverse workloads and applications, a one-fit-all pricing policy does not provide flexibility to the user. Energy efficiency, cost, and profit are interdependent. A flexible way of controlling cloud systems is proposed in paper [27] to satisfy the user and energy cost.

The bidding strategies based methods are mainly based on cost-benefit analysis [31]. Here VM instances are allocated to the customers based on all the received bids, as well as on the current available computing capacity. The bid value above or below this published price is declared either successful or unsuccessful. Auction based methods depend on the spot instance price of the resources. Users can submit bids to the market at any time, using the spot price history to decide how much to bid. The provider sets the spot price at regular time intervals, e.g., every five minutes, depending on the number of bids received from users (demand) and how many resources are available (supply) at each time slot [28, 32]. In these mechanisms, users’ bids above the spot price are accepted, and that below is rejected in each time slot. Running spot instances [37] are terminated if their original bid prices fall below the new spot price and re-launched only when their bids again exceed the spot price. Usually, these sport prices are based on historical prices.

The explosive growth of the Internet of Things (IoT), big data, and emerging fog computing makes the involved services and related resource management makes more complicated than ever before. Due to resource limitations [32], resource heterogeneity [33], locality restrictions, environmental necessities and dynamic nature of resource demand, resource allocation and scheduling are one of the essential problems, to be taken into account to adapt to the changing infrastructure environments [35]. The current literatures give only an overview and no substantive research on the above issues.

Table 2.4: Minimize cost

Paper	Method	Parameters	Highlights	Limitations	Environment
[27]	Dynamic replication	Cost	Reducing horizon control	No load balancing	Simulation
[28]	Offline simple task scheduling	Cost Makespan	Cost optimality Cost performance tradeoffs	Slow performance	Simulation
[29]	Dynamic Data Allocation	Cost Makespan	Reduction in operational cost	No load balancing	Simulation
[30]	Spot and dynamic pricing	Cost Resource use Waiting time Interruption rate	High biding option in online market	Performance overhead	Simulation
[31]	Bidding strategy	Spot price Bid price	Optimal bidding	Interruption overhead	Simulation
[32]	Multi-criteria decision making framework	Cost/benefit ratio	Reduce execution time	No load balancing	Simulation

Table 2.4: Minimize cost (Continued…)

Paper	**Method**	**Parameters**	**Highlights**	**Limitations**	**Environment**
[33]	Bayes classifier	Cost Waiting time Deadline	Minimize execution time Minimize operational cost	Interruption rate is high	Simulation
[34]	Priced Timed Petri Net (PTPN)	Completion time Cost	Pre-allocated resources Credibility evaluation	No load balancing	Simulation
[35]	Scheduling based on Credit and Cost	Cost Task penalty Credit price	Discriminating function Maximization of service supplier	Interruption overhead No fairness among tasks	Simulation
[36]	Paddy Field Algorithm (PFA) Price detection algorithm	Cost Execution Time	Combinatorial double auction policy Better service satisfaction	Need balancing of bid price and spot price	Simulation
[37]	Holistic brokerage model	Cost	Scalability SLA negotiation\	Underutilization of resources	Simulation

In the Petri Net model [34] credit evaluation about a provider is taken as the primary parameter for task allocation. This uses an income discriminate function value as a decision making factor for task pre-emption. This market scheduler first schedules service-suppliers' tasks with worse credibility among users while realizing the income maximization of service suppliers so as to eradicate their bad impression of "income-oriented", but it doesn't employ load balancing methods.

Table 2.5: Maximize profit

Paper	Method	Parameters	Highlights	Limitations	Environment
[239]	Mixed Integer Non Linear Programming (MNLP) formulation	Profit Service Penalty	Server consolidation Heuristic method	High SLA violations Slow No load balancing	Simulation
[240]	Price detection algorithm Cooperation Competition	Revenue Profit	Minimum energy consumption Revenue maximization	Network latency Delay No load balancing	Real
[241]	Profit driven optimization	Profit Execution Time	Scalability	Delay in service Low makespan	Simulation

Market oriented cloud is another model to provide high QoS to the customers and manage this quality during its lifetime [37]. Here providers have to consider the different service quality parameters of each customer. Here cloud resource management is based on the supply demand ratio of resource and the aim is to reach market equilibrium [29, 36, 37]. Cost minimization methods are summarized in table 2.4.

Virtual Network Monitoring (VNM) is a big challenge for the service provider since users send numerous requests to reserve computational and network resources and they expect their QoS conditions to be maintained through the request lifetime [38]. Price detection algorithm [39], profit-driven optimization [40] are mainly focused on profit but these methods have high SLA violations. The merits and demerits of profit-oriented methods are given in table 2.5.

2.2.5 Energy

The articles [38, 39] are based on Dynamic Voltage and Frequency Scaling (DVFS) is proposed to reduce energy consumption. PreAnt [40] policy with Bin packing algorithm also focused to reduce the energy consumption. All the above methods lack QoS support thus low efficiency. In paper [41] service providers and users reached an agreement on energy-aware scheduling services. The collaborative approach mentioned in [42] is also another such approach. The summary of the above methods is given in table 2.6.

2.2.6 Priority

Some researchers considered priority parameters to schedule the tasks but priority consideration is only good for high performance scientific computing. The Memetic Algorithm (MA) in [63] merges the concept of local and population based search to find a solution to the scheduling problem. It is a static task scheduling scheme and not suitable for a dynamic cloud environment. Another method [64] is based on multiple priority queues. In a cloud computing environment, multiple customers are submitting job requests with their constraints. This method is suitable for scientific simulations such as weather prediction, rainfall simulation, Monsoon prediction, and cyclone simulation, etc., requires a huge amount of computing resources such as processors, servers, storage, etc. In this situation, it will be a problem for the cloud administrator to decide how to allocate the available resources among the requested users to minimize makespan and utilize resources effectively [65]. The summary table for the above methods is provided in table 2.7.

Table 2.6: Energy

Paper	Method	Parameters	Highlights	Limitations	Environment
[38]	DVFS Bin packing algorithm	Execution time Cost Energy Frequency	Minimum energy consumption ratio (ECR) Minimum worst-case execution time (WCET)	Low efficiency	Simulation
[39]	DVS Energy-aware Dynamic Task Scheduling (EDTS)	Energy Execution time Cost	Minimum energy consumption Reduce cost	Lack of QoS support No load balancing	Simulation
[40]	PreAnt policy Bin packing algorithm	Energy Execution time	Manage instantaneous peak load Resource intensive application with QoS	Service interruption	Simulation
[41]	Optimal resource allocation with pre-determined task placement & resource allocation algorithm	Energy Cost Job completion time	Increase utility and productivity Linear programming method	No load balancing Perform-ance degradation	Simulation
[42]	Lagrange relaxation based Aggregated Cost Algorithm (LRAC)	Energy Delay Deadline	Collaborative task execution One-climb policy Minimum energy consumption	Low efficiency No load balancing	Simulation

Table 2.7: Priority

Paper	Method	Parameter	Highlights	Limitations	Environment
[63]	Memetic - GA method	Makespan Speed	Optimization Earliest finishing time	Delay No load balancing	Simulation
[64]	Priority algorithm	Time Cost	Maximum profit Minimum wastage of resources	Frequent migrations Low response time	Simulation
[65]	Min-Min algorithm Priority-based scheduling	Makespan Cost	Scalability Load balancing	Less fault tolerance Frequent migrations	Simulation

2.2.7 Multi objective

Multi-objective task scheduling algorithms are predominantly well suited to deal with cloud optimization problems. Some meta-heuristics methods like simulated annealing [46], Evolutionary Algorithms (EA) [47] and Particle Swarm Optimization (PSO) [43] have been proposed to address resource allocation process in the cloud. Since Genetic Algorithm (GA) is highly time complex, it is not practically suited for large-scale applications. A PSO-based heuristics [44] is another method to schedule applications to cloud resources that take into accounts both computation and energy cost. The main limitation of evolutionary algorithms is their high computational cost due to their slow convergence rate. So some sort of hybridization or enhancement is needed in this type of method. The DVFS based method [43] is to minimize energy consumption along with time and cost. This technique allows processors to operate in different voltage supply levels by sacrificing clock frequencies.

The main aim of multi-objective scheduling strategy is to find a trade-off between customer requirements and provider or resource constraints. i.e., the user-submitted tasks have different requirements on computing time, memory space, data traffic, deadline, response time, etc. While the cloud resources are heterogeneous and distributed. One of the problems in the meta-heuristic method is the ability to avoid getting stuck with sub optimal solutions. Most of the nature inspired heuristic algorithms like GA and bee colony [45] are suitable for cloud scheduling, but we have provided detailed reviews in section 2.6. The summarized information about multi objective methods are shown in table 2.8.

Table 2.8: Multi-objective

Paper	Method	Parameter	Highlights	Limitations	Environment
[43]	PSO, DVFS & HEFT algorithm	Cost Time Energy	Workflow Scheduling Energy consumption	Low efficiency Sub optimal response time	Simulation
[44]	Nested PSO-based multi-objective task scheduling	Energy time	Energy optimization	Low service availability Frequent migrations	Simulation
[45]	ABC Algorithm	Cost Time Energy	Optimization in time and cost	Frequent migrations	Simulation
[46]	Multi-objective cat swarm optimization with SA	Time Cost	Scalability	Low efficiency Slower Sub optimal solutions	Simulation
[47]	Multi-objective Evolutionary Algorithm (MEA)	Waiting time Cost Energy	Minimize energy consumption Cost and time optimization	Low efficiency Slower Sub optimal solutions	Simulation
[48]	Min-Min based time and cost trade-off algorithm	Time Cost	Multi-objective optimization model	Lack of failure recovery	Simulation

2.3 VM Placement Methods

In IaaS cloud, the major interface to the users to run their applications through VMs. Here the users can create or maintain with their own VM preferences. Also, they can maintain software installations and have complete control over their VM Images. VM creation and management is complex due to its scale and variety. The VM image content can be stored as a file, a block device, a logical volume, a root partition or a complete hard disk drive. So VM placement is a big challenging problem.

Table 2.9: VM placement

Paper	Method	Parameter	Highlights	Limitations	Environment
[49]	Common Deployment Model (CDM)	Time Bandwidth Memory	Maximize resource utilization Use of active and passive directory	Unable to handle network latency	Simulation
[50]	Adaptive spread based scheduling algorithm	Bandwidth Cost Response time	Slicing scheduled tenant request model Maximize acceptance rate Minimize power usage rate	Low efficiency Slow Low response time	Simulation
[51]	Discrete PSO	Response time Cost	Maximize resource utilization Minimize energy consumption	Less reliable Low response time	Simulation
[52]	MigrateFS algorithm	Cost Execution time	Optimization model Scalability Detecting SLA violation	Low performance	Simulation
[53]	VM resource dynamic scheduling algorithm	Price Bandwidth	Resource utilization Minimize pricing	Low performance No load balancing	Simulation
[54]	Greedy & PSO Algorithm	Completion Time Cost	Convergence rate is optimized Reduced completion time	No load balancing	Simulation
[55]	PSO	Energy	Energy efficient VM placement	No load balancing No SLA	Simulation
[56]	Improved PSO	Time	Increased resource availability	No load balancing No SLA	Simulation
[57]	Hybrid discrete PSO	Cost Energy	Energy efficient VM placement	Frequent migrations	Simulation

The adaptive spread policy based on PSO algorithm used [51] is one such method to differentiate the long and short user requests based on a threshold value. It achieves energy savings and carbon emissions reduction, using server consolidation technology. It consolidates multiple applications on the same physical machine, with each application typically running on its own virtual machines. In the context of virtualized data centers, it is a critical concern to design energy-efficient virtual machine placement approaches that reduce energy consumption while satisfying customers [55, 56, 57].

The bandwidth oriented mechanism [53] also load balancing issue. In short, we can say that there are two VM placement models namely conventional and economic models. The conventional models assume that resource providers are non-strategic, whereas economic models assume that resource providers are rational and intelligent. In conventional methods, a user pays for the consumed service. In economic models, a user pays are based on the value derived from the service. Hence cost-aware VM placement models are more appropriate in the context of cloud. The details are summarized in table 2.9.

2.4 Load Balancing Methods

An optimal load in each physical server will improve the system performance. A load balancing method aims to avoid overloaded or under loaded conditions in a physical machine or server. Too much load will result in the violations in SLA conditions and thereby performance degradation and financial loss. So to maintain QoS guaranteed service providers have to adopt suitable load balancing mechanisms across their computational resources. When overload causes performance degradation while under loaded conditions will results in high power consumption, energy and cost. An advanced cross-entropy based stochastic method [58] for workload scheduling proposed load balancing is one such method but it creates frequent migrations. Developing high performance workload scheduling techniques in cloud computing imposes a great challenge that has been extensively studied by several researchers. Most of the previous works aim only at minimizing the completion time of tasks. However, timeliness is not the only concern, while reliability and security are also very important. The load-balanced scheduling focuses on evenly distributing traffic among all links in a data centre network to enable the network to transmit more data flows with lower average end-to-end transmission delay.

Due to high cost and low programmable ability, traditional hardware based load balancing techniques cannot be widely used in datacenters. Therefore, some researchers pay more attention on software-defined networking (SDN) techniques (e.g., OpenFlow) [59] that can improve the transmission capacity of data centers through programmable load balanced flow control.

A Task Based System Load Balancing method using Particle Swarm Optimization (TBSLBPSO) [60] that achieves system load balancing by transferring only extra tasks from an overloaded VM instead of migrating the entire overloaded VM. There are several other models to migrate and balance workload across data centre to improve computation [61]. A good load balancing method also has to limit frequent migrations. Frequent migrations will create an imbalance in the system, and that ultimately affects performance. We have reviewed several literatures that deals with load balancing issue and the summary of the findings of these methods are tabulated in table 2.10.

Table 2.10: Load balancing methods

Paper	Method	Parameter	Highlights	Limitations	Environment
[58]	Advanced Cross-Entropy based Stochastic Scheduling	Service rate Arrival rate	Scalability Flexibility Optimize QoS	Delay Frequent migrations	Simulation
[59]	Static offline optimal algorithm Network Overhead Minimization	Bandwidth	Minimize inter-datacenter network load reduction	Low efficiency Delay	Simulation
[60]	Task-Based System Load Balancing (TBSLB)	Time Transfer time Cost	Pre-copy process maximizes resource consumption	Delay Frequent migrations	Simulation
[61]	Two stage load balancing	Cost Power	Pareto optimality	Low performance Delay	Simulation

2.5 Dynamic and Adaptive Methods

Dynamic and adaptive methods are needed to schedule the diverse and distributed cloud resources efficiently. In order to meet customer requirements, this kind of method is necessary for the rapid and efficient leverage of cloud resources.

2.5.1 SLA aware

The aim of SLA aware methods is to provide services with high-quality service as customer requested. To harmonize the SLA as well as to reduce operational cost intelligent mechanisms are needed. Only a few SLA aware works are currently available in this area. An SLA aware hybrid cloud scheduling algorithm is used in an elastic autonomous service network to solve these issues [62]. The details are shown in table 2.11.

Table 2.11: SLA aware

Paper	Method	Parameter	Highlights	Limitations	Environment
[62]	Hybrid cloud scheduler algorithm	Cost Deadline	Elastic autonomous service network	No load balancing	Simulation
[189]	Power aware consolidation	Cost Power	PM Clustering	Limited to scientific workflows	Simulation
[231]	Elastic service placement	Cost	Column generation method	Sub optimal solution	Simulation

2.5.2 Elasticity based

In the cloud, elasticity can be defined as how the amount of computing resource changes with the current workload. This definition is quantitative and measurable; however, such a definition of responsiveness is not entirely adequate, since it only considers how much, not how fast, the computing resource adapts. If the provider takes a long time to provide the correct amount of resources to match the workload (which might not be current anymore), it is not considered as elastic. So the elasticity is meaningful to the cloud users only when the acquired VM resources can be provisioned in time within the user expectation. The long unexpected VM start-up time could result in resource under-provisioning, which will inevitably hurt system performance [67]. Similarly, the long unexpected VM shut-down time could result in resource over-provisioning, which will inevitably hurt resource utilization. The auto-scaling capability of the cloud can ensure the service with QoS with minimizing the makespan and cost [68]. Table 2.12 gives a summary of elasticity based methods in cloud scheduling.

Table 2.12: Elasticity based

Paper	Method	Parameter	Highlights	Limitations	Environment
[66]	Open Cloud Computing Interface (OCCI)	Time Cost	Autonomic loop	Multiple autonomic loop	Real
[67]	On-site elastic algorithm	Execution time Cost	Multi-level QoS service	Performance degradation & Delay Frequent migrations	Simulation
[68]	Dynamic Fault-Tolerant Scheduling (FASTER) Algorithm	Execution Deadline	Primary backup-based scheduling Auto scaling Backward shifting Resource utilization	Delay No load balancing	Simulation

2.6 Optimization Methods

As mentioned in the introduction, another classification of scheduling method is based on the optimization policies used in the algorithms. The dynamic nature of the cloud environment makes task scheduling as a cumbersome task. Scheduling in the dynamic cloud environment is NP-hard, so finding an optimal solution for the task assignment is difficult. Also, the solutions are obtained by taking several assumptions on the state of the cloud ecosystem. Nature inspired algorithms are capable to produce good sub optimal solutions using heuristics. Heuristics used by ants, bees, and flock of birds are some of the examples. The sub optimal category of algorithms can be further classified into heuristic, meta-heuristic and hybrid algorithms, based on how they are applied in the application scenario.

2.6.1 Linear programming model

Linear programming (also called linear optimization) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. It is a special case of mathematical programming. An intelligent agent based approach [69] considers availability, price and time as scheduling criterion, but it lacks load balancing. Large scale cloudlet scheduling mechanism proposed in [70] is based on bandwidth and latency, but it is only suitable for scientific workloads and suffers a load balancing issue. A fault-tolerant system with less power consumption is created by a Bayesian approach [71] is good, but it is to be improved to consider load balancing and cost. Stochastic models [72], RIAL [73], Greedy method [74] produces only near optimal solutions. The summary of the above methods is shown in table 2.13.

Table 2.13: Linear programming models

Paper	Method	Parameter	Highlights	Limitations	Environment
[69]	Intelligent agent based approach	Price Availability Time	Agent-based computing Event condition action	No interoperability No load balancing	Simulation
[70]	Optimum cloudlet selection strategy	Latency Bandwidth Response time	Large scaling of cloudlet deployment Optimal cloudlet placement	No workload management No load balancing	Real
[71]	Bayesian Approach Semi-Markov model	Energy Execution time	Fault recovery system	No power consumption No cost consideration	Simulation
[72]	Hierarchical Stochastic modeling	Time Workload	Workload management	Execution cost is high No load balancing	Simulation
[73]	Resource Intensive Aware Load (RIAL) Balancing	Bandwidth Memory Time cost	Minimize VM communication cost Load balancing	Sub optimal solutions Frequent migrations	Simulation
[74]	Greedy algorithm	Time	Revenue maximization	No SLA No power consumption	Simulation
[76]	Practical outsourcing	Cost overhead	Security	Frequent migrations	Simulation
[77]	Integer Linear Programming (ILP)	Power, cost Storage Bandwidth	Machine learning based VM allocation	High overhead	Simulation
[78]	Duality Theorem Affine Mapping	Cost	Feasible region protection	No stability	Simulation

2.6.2 Heuristic methods

Heuristic methods are another class of methods proposed for cloud scheduling. The term heuristic is used for algorithms that find solutions among all possible ones, but they do not guarantee the optimal result. So they can be considered as approximate algorithms. These algorithms, usually find a solution close to the best one and they find it fast and easily. These algorithms are designed to solve problems in a faster and more efficient manner than traditional methods by sacrificing optimality, accuracy, precision or completeness for speed.

There are few articles that discuss heuristic algorithms which suggest some approximations to the solution of optimization problems. In such problems, the objective is to find the optimal of all possible solutions by minimizing or maximizing the objective function [80, 83, 85, 86]. In this method, the objective function is used to evaluate the quality of the generated solution. Even if an exact algorithm can be developed, its time or space complexity may turn out unacceptable. In reality, it is often sufficient to find an approximate or partial solution. Such admission extends the set of techniques to cope with the problem. Heuristic methods are covered in table 2.14.

2.6.3 Meta-heuristic methods

Heuristic algorithms are good for specific applications and it gives optimal solutions within a specific time. Meta-heuristic algorithms are computationally more complex than heuristic algorithms and more suited for general purpose problems. In the dynamic cloud, the environment is challenging, meta-heuristics are a good solution for obtaining optimal solutions. Several cloud scheduling methods that used meta-heuristic approach are based on nature inspired algorithms. The most prominent nature inspired methods used for cloud scheduling are shown in figure 2.4.

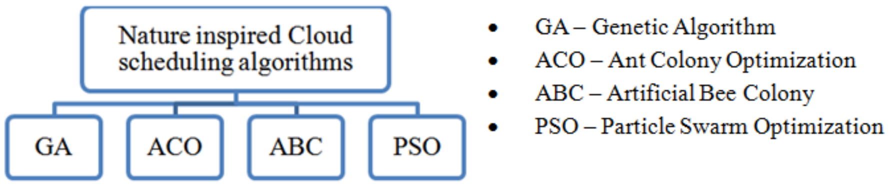

Nature inspired Cloud scheduling algorithms
GA
ACO
ABC
PSO
GA – Genetic Algorithm
ACO – Ant Colony Optimization
ABC – Artificial Bee Colony
PSO – Particle Swarm Optimization

Fig. 2.4 Nature inspired algorithms

Table 2.14: Heuristic methods

Paper	**Method**	**Parameter**	**Highlights**	**Limitations**	**Environment**
[79]	Greedy-Ant scheduling	Makespan Priority	Minimize execution time	Slow	Simulation
[80]	Modified Best-Fit Decreasing (MBFD)	Cost Energy	Autonomic energy-aware mechanism	No SLA	Simulation
[81]	Elasticity Based Scheduling Heuristic (EBSH)	Cost Profit	Self-managed	Inefficient Slow	Simulation
[82]	Local search	Energy Time Bandwidth	Minimize energy consumption	No load balancing	Real
[83]	Based on Bayes theorem and Clustering	Cost Makespan	Maximize posteriori probability value	Low throughput	Simulation
[84]	PSO algorithm	Cost	Distribution of workload	No energy consideration	Simulation
[85]	Critical-Path based heuristic	Execution time Cost	Good time management	Frequent migrations	Simulation
[86]	Hyper-Heuristic Scheduling Algorithm (HHSA)	Cost Makespan	Optimization in makespan	Frequent migrations	Simulation

2.6.3.1 Genetic Algorithm

The basis of Genetic Algorithm (GA) is the principle of evolution and natural genetics. It combines the exploitation of past results with the exploration of new areas of the search space. By using the survival of the fittest techniques combined with a structured yet randomized information exchange, GA can mimic some of the innovative flair of human search [87]. Genetic algorithms based cloud scheduling shows great efficiency in small instances as in timetabling problems, but are not efficient in large instances. GA combined with the stochastic method also shows low efficiency [88]. The tabular information about GA based methods is given in table 2.15.

Table 2.15: GA based methods

Paper	Method	Parameter	Highlights	Limitations	Environment
[87]	GA Local Search (LS) technique	Completion Time/Makespan Workload	Minimize completion time	Sub optimal solutions	Simulation
[88]	Johnson's rule based GA	Makespan Cost	Multi-processor scheduling Low complexity	No load balancing	Simulation

2.6.3.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is based on real ant's behaviour to find a good food source from their nest. The principle behind ACO based algorithm is that ant's ability to produces pheromone and leaves it into the way they travel. The intensity of pheromone increases when more ants travel on the same way. Then find out the shortest path based on the intensity of the pheromone. The ACO based scheduling methods simulate the searching behaviour of artificial ant's colonies to find a solution. The papers [90, 91, 95] proposed ACO to reduce the makespan of tasks. While methods proposed in [92, 93, 94] are to reduce energy consumption in the cloud datacenters. A few methods are derived to load balance the cloud [96]. The convergence speeds of these algorithms are quite slow, hybrid methods are necessary to speed up and to deal with multi-objective optimization. ACO based methods are tabulated in table 2.16.

Table 2.16: Ant Colony Optimization methods

Paper	**Method**	**Parameters**	**Highlights**	**Limitations**	**Environment**
[89]	Basic ACO	Makespan	Random optimization	No load balancing Slow	Simulation
[90]	Modified ACO	Response time Throughput	Two level cloud scheduler	High network communication Slow	Simulation
[91]	Load balanced ACO	Makespan	Load balancing	Slower when number of iterations are high	Simulation
[92]	Basic ACO	Energy	Energy aware	No load balancing	Simulation
[93]	Modified ACO	Energy	VM consolidation	No load balancing	Simulation
[94]	Multi objective ACO	Energy Resource usage	Scalability	No load balancing	Simulation
[95]	List ACO	Deadline Cost	Deadline constrained	Slow	Simulation
[96]	LB-ACO	Makespan	Load balancing Multi-objective Scheduling	Sub optimal solutions	Simulation

2.6.3.3 Artificial Bee Colony methods

Artificial Bee Colony (ABC) algorithm is developed based on the foraging behaviour of honey bees to find a food source. It can be compared with other methods. The algorithm gives the efficient performance as it uses both global exploration search and local exploitation search. The works in [97] addressed time and cost but not considered load balancing issues. While the paper [98] tested in private cloud system focused only on energy consumption. The inefficient load balancing mechanism used in heuristic ABC [99] causes frequent migrations. Our findings are summarized in table 2.17.

Table 2.17: Artificial Bee Colony methods

Paper	Method	Parameter	Highlights	Limitations	Environment
[97]	Pareto- based ABC	Response time Cost Makespan	High profit Minimize cost Load balancing	No priority Frequent migrations	Simulation
[98]	Power-aware ABC	Power Energy	Energy consumption	Delay No load balancing	Simulation
[99]	Heuristic ABC (HABC)	Makespan Cost	Maximize resource utilization Load balancing	Inefficient load balancing	Simulation

2.6.3.4 Particle Swarm Optimization methods

Continuous optimization without prior information is the principle behind Particle Swarm Optimization (PSO). Researchers have proposed several methods to address cloud scheduling problems with multiple objectives [102, 106, 109]. The PSO-based methods [55, 56, 57, 109, 114] balance the load across the data centres. While the methods [108, 110, 113] focused on computation time, deadline, energy, and profit without load balancing. Table 2.18 gives a summary of PSO methods.

Table 2.18: Particle Swarm Optimization methods

Paper	Method	Parameter	Highlights	Limitations	Environment
[100]	PSO	Makespan Execution time	Optimized execution time	No QoS	Simulation
[101]	Modified PSO GA	Completion time Makespan	Load balancing Minimized Execution time	Slow	Simulation
[102]	MOPSO	Makespan Waiting time	Minimum time & energy	No load balancing	Simulation
[103]	PSO	Execution time Response time Cost	Lower execution time	No scalability	Simulation
[104]	Self- adaptive learning PSO	Makespan Cost	Load balancing based on resource usage	No SLA	Simulation
[105]	PSO	Makespan	Minimizes VMs down time	No SLA	Simulation
[106]	Multi-objective Pareto based PSO	Makespan Cost	Dynamic voltage and frequency scaling	SLA and energy not considered	Simulation
[107]	PSO for Energy Saving (PS-ES)	Energy Time	Self adaptive Minimize energy	Homogeneous cloud Higher migration rate	Simulation
[55]	PSO	Energy	VM placement	No load balancing No SLA	Simulation

Table 2.18: Particle Swarm Optimization methods (Continued…)

Paper	Method	Parameter	Highlights	Limitations	Environment
[56]	Improved PSO	Energy	VM placement	No load balancing No SLA	Simulation
[57]	Hybrid PSO	Cost Energy	Energy efficient VM placement with PSO-TS	Slow Frequent migrations	Simulation
[108]	Self-Adaptive Learning PSO	Deadline Cost	No formal inter-cloud agreement is needed to outsource tasks	No load balancing	Simulation
[109]	Multi-objective PSO	Time Energy	Considered scheduling problem as a discrete task permutation	Only quasi-optimal solutions No load balancing	Simulation
[110]	Heterogeneous dynamic resource provisioning	Deadline Cost	Minimize overall execution cost while meeting a user defined deadline	Convergent time is high Slow	Simulation
[111]	PSO	Cost Time	PSO with embedded cross over and mutation operation	Energy and SLA not considered	Simulation
[112]	PSO	Computation Transmission cost	Simple heuristics PSO with load consideration	Energy and SLA not considered	Simulation
[113]	Discrete PSO	Cost Deadline	Discrete PSO with deadline constraints	No load balancing No SLA	Simulation

2.6.4 Hybrid methods

Hybrid methods are a combination of two or more algorithms to perform a task and obtain optimal solutions than a single algorithm. These algorithms are suitable for NP-hard problems like cloud scheduling in a cost effective manner with minimum execution time.

Table 2.19: Hybrid methods

Paper	Method	Parameter	Highlights	Limitations	Environment
[114]	Hybrid PSO	Makespan Cost Imbalance	List based heuristic algorithm	No SLA	Simulation
[115]	SA-PSO Temporal delay bound	Delay Cost	Optimized throughput Delay bound	No QoS	Simulation
[116]	ACO-ABC-PSO Dynamic meta-heuristic	Makespan Cost Energy	Load balancing	No QoS	Simulation
[117]	ACO-PSO-SA Scalable multi-objective-Cat Swarm Optimization based SA (CSM-CSOSA)	Makespan Cost Energy	Load balancing Reduce operational cost	Slow Frequent migrations	Simulation
[118]	ACO-PSO Hybrid meta-heuristic	Response time Resource utilization	High fault tolerance High resource utilization Low computing time under high load Low response time	Homogeneous servers. High cost	Simulation
[119]	Hybrid ACO-PSO	Resource utilization Makespan	Avoids premature solutions	Single objective No load balancing	Simulation
[120]	ACO-PSO with Min-Max	Makespan Cost	Load sharing	Single targeted scheduling No SLA	Simulation
[121]	GA-PSO GA - Hybrid PSO method	Makespan Cost	High resource utilization Low computing time	Low efficiency No SLA	Simulation

Temporal Task Scheduling Algorithm (TTSA) is an example of optimizing the throughput by using hybrid methods [115] in the cloud. It considers delay and cost factor but SLA factors not considered. To improve the efficiency of the scheduling process, currently available literatures are considering different parameters. Cloud Scalable Multi-objective (CSM) task scheduling and optimization algorithm [117] based on Simulated Annealing (SA) algorithm considers execution time and cost. The novelty of the method is that its design enhances the local search procedure of the algorithm in exploring a larger search space that returns better optimum solutions. It is slow and frequent migrations affect system performance.

Nature inspired algorithms can easily combine with classical algorithms or with other heuristic algorithms, which gives better results. The hybrid algorithms mentioned here are based on response time [118], artificial intelligence network load balancing using ACO [119] and modified GA [120].

In paper [121] the crossover strategy and mutation strategy of the GA is embedded into PSO, so that it can play a role in the discrete problem. This hybrid method improves the solution quality, so it can be used as an effective way to solve the cost minimization problem in workflow scheduling, but convergence speed is low. Table 2.19 summarizes the different hybrid methods in cloud resource and task management.

2.7 Review Observations

Our detailed literature review analyzed various problems in resource allocation, task scheduling, VM placement, and load balancing methods in the cloud. These literatures are grouped based on the objectives considered, methodology used, etc. Also, we have analyzed the platform in which the methods were tested.

Since the cloud is a business model, financial considerations are the primary issue addressed by most of the methods. Service providers always look for profit by maximization of their resource utilization and minimization of operational cost, energy, while consumer focuses on better quality oriented service within minimum cost and time. Other observations are

- **Makespan Minimization:** One of the important parameters that directly affect QoS is the makespan, which needs more attention.
- **System Stability and Load Balancing:** Methods proposed for load balancing affect system stability severely due to frequent migrations. This needs immediate attention to achieve QoS in cloud scheduling.
- **Energy conservation:** In order to harness the green computing data centers need energy-aware resource allocation methods.
- **SLA consideration:** Guaranteeing SLA is the key task in maintaining good quality of service.
- **Optimization methods:** Cloud scheduling is multi-objective optimization problem with conflicting objectives. Most of the methods describe in the literature tested in static conditions and major consideration is a single parameter. Integrated QoS scheduling methods are needed due to the dynamic nature of the cloud environment.

2.8 Design Considerations of the Thesis

An optimal scheduling policy is to be designed to mitigate the issues in cloud scheduling and ensure QoS. Hence, the design considerations are:

Optimal scheduling to minimize makespan: The improvement in the efficiency of the scheduling process mostly in terms of makespan, cost, and profit. The optimal scheduling improves cloud performance, by minimizing makespan, operational cost and response time. This benefits both customers as well as cloud service providers. Other factors that affect QoS in cloud scheduling needs are to be addressed. For these advanced techniques are needed to consider these factors to improve QoS in the cloud.

Load balancing and VM placement to achieve better makespan and stability: Proper placement of workload across multiple physical servers will enhance the system performance in terms of makespan and stability. Since the frequent VM migrations affect system stability, an enhanced load balancing and VM placement is required.

Energy consideration: Since cloud datacenters consume a large quantity of power, intelligent power aware resource monitoring and managing methods are needed to support green computing.

Cost and budget control: In the pay-per-use paradigm, the resources and services are being billed per usage, so changes in computation cost is a vital factor in adopting cloud computing. Hence a cost aware-budget control system is needed for transparency.

Scalability: SLA violations can be reduced with dynamic autoscaling of resources. The energy, delay, deadline, time and cost affect the scalability. Dynamic methods are needed for proper scaling decisions.

QoS and SLA: SLA oriented computing promises services with certain quality conditions stipulated in the agreement between customers and providers. Even though the quality of a service depends on customer perception and quality of experience, intelligent methods are necessary to manage and ensure QoS. Efficient QoS and SLA oriented methods will also reduce violations in the SLA.

Prediction mechanisms: Efficient workload prediction mechanisms to conduct service level violation free resource allocation. An interference prediction mechanism to mitigate SLA violations and aid auto scaling.

2.9 Metrics

Different performance metrics to evaluate the effectiveness of the methods proposed in this thesis are as follows:

1. Makespan
2. Number of Physical Machines used
3. Energy/Power
4. Number of VM Migrations
5. SLA Violations

Apart from this, specific metrics are used for evaluation purposes in the respective chapter of the thesis.

2.10 Summary

This chapter reviewed and identified highlights and limitations of different scheduling methodologies. After careful study, we have decided to address the issues in cloud scheduling to provide better QoS. Also, various metrics are identified to evaluate the performance of the proposed methods.

CHAPTER 3
HANDLING MAKESPAN

Contents

3.1 Introduction

In this chapter, a VM placement mechanism is proposed for the improved quality in the cloud scheduling by handling makespan. This technique is based on the principle of Bin packing method. The main objective of the proposed work is to minimize the makespan, maximize the cloud resource utilization as well as the reduction in power consumption. We have implemented four methods namely

Best-Best, Worst-Worst, Worst-Best, and Best-Remaining. In the proposed Best-Remaining method the incoming user requests are scheduled using the best-fit method. The cloud broker employs worst-fit method for VM placement. The experimental results show that the Best-fit – Remaining-fit strategy reduces makespan compared to other methods. It also efficiently places the VMs to the less number of active physical servers that improves the performance of the cloud system.

3.1.1 Makespan and VM placement

The most critical operation in cloud scheduling is VM placement. VM placement is the process of determining the most appropriate physical server or machine to host the user requested VM. The optimal selection of the physical server may be based on makespan, energy consideration, processing power, and resource utilization. These VM selections should also consider QoS parameters provided by the service provider and requested by the cloud customer. Usually, all users are concerned about the makespan of their submitted tasks [15].

In IaaS cloud main interface to the cloud resources are VMs where users run their applications. Depending upon the provider policy, sometimes they allow users to create and maintain VM images (VMI) with their own requirements (e.g., on Amazon EC2). The design and implementation of the virtual machine image management mechanism are challenging, due to the scale, complexity, variety, and dynamics of VMIs. Since cloud providers are more concerned about energy reduction, carbon emission reduction, optimal VM placement and server consolidation mechanism is needed. An efficient placement mechanism allocates the optimal number of VMs in a physical server to maintain makespan consideration. So it is a critical task to design an energy-efficient VM placement mechanism with other quality of service requirements especially makespan requirement by the customer [55, 57, 122]. The overview of VM placement mechanism is shown in figure 3.1. Here physical servers are sliced into a number of VMs. The user requested VMs are mapped into the appropriate VM images available in the service provider's conditions. Then the VM placement mechanism maps these VM images to the physical servers based on the current status of each server. These conditions may include load, makespan, storage and memory requirements, etc.

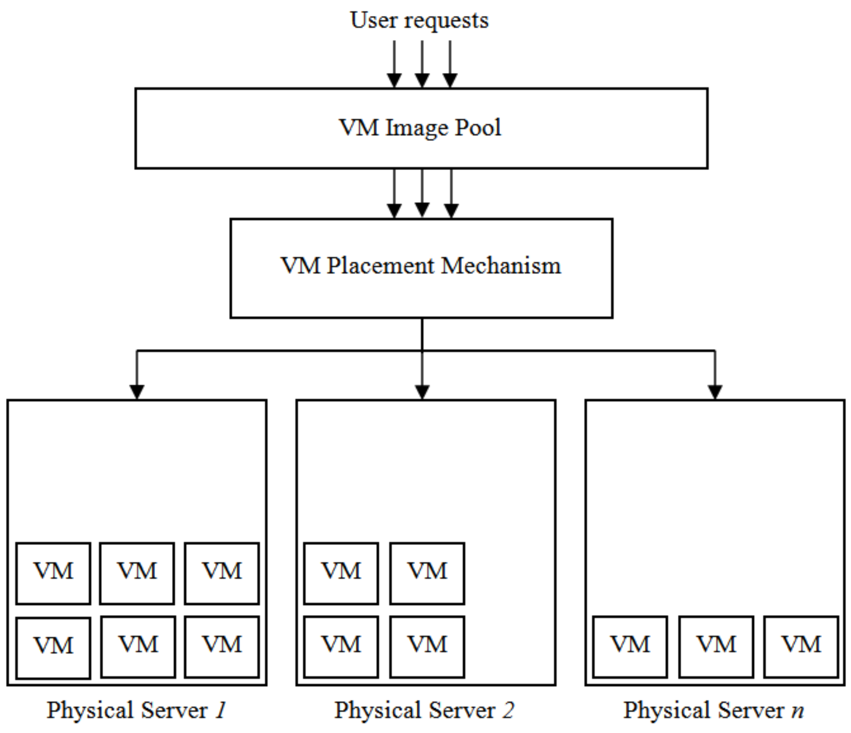


Fig. 3.1 Overview of VM Placement Mechanism

The resource multiplexing with the help of virtualization technology improves the overall utilization rate and mainly reduces the total cost of ownership. Several works are available to solve the issues in VM placement in the cloud, which addresses various performance parameters [123], availability [124], network [125], and cost [126]. The service providers are more concerned about revenue generation in VM placement [127, 128]. Providers always tried to find an optimal VM placement that will minimize the cost of operation and maintenance of the infrastructure with good quality of service to the users.

3.2 Proposed Method

Our proposed architecture consists of the following components: cloud customers, cloud broker, database manager, virtual machines, physical machines, and several cloud providers. The architecture is shown in figure 3.2.

Here the jobs are assigned to the physical servers in two stages. In the first phase, the customers submit their jobs and respective requirements to the service providers. This is done through the cloud broker. The cloud broker acts as an intermediary component between the customers and the providers. The responsibility of the broker is to place the submitted jobs to the appropriate VMs provided by the service providers. The VM repository stores predefined images of the VMs. In the second phase, the optimization of VM placements to the physical machines are done.

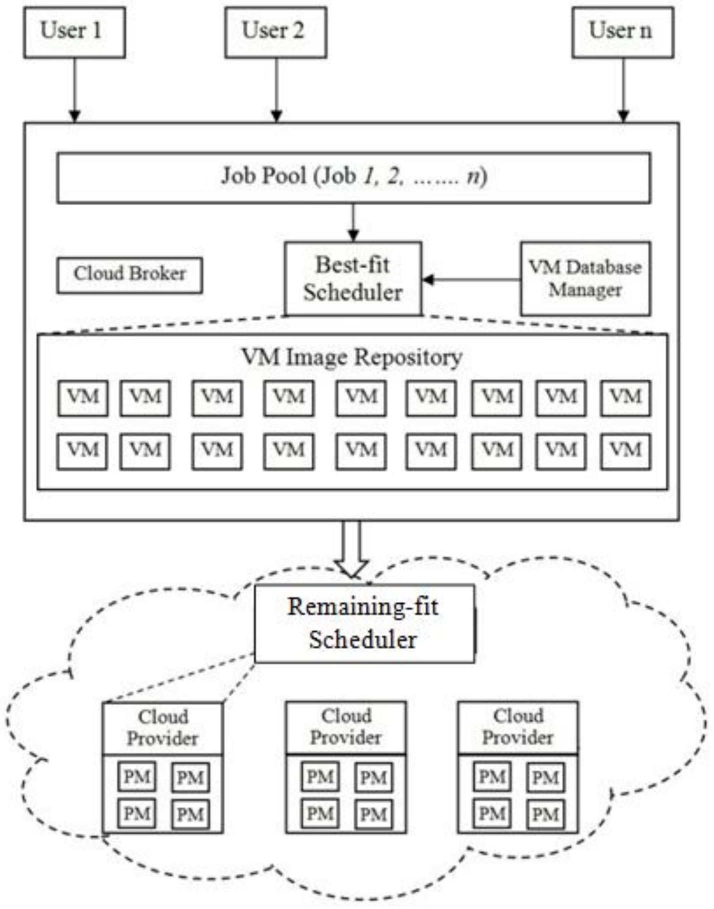


Fig. 3.2 Cloud Architecture

3.2.1 Optimal allocation

For the optimal performance dynamic migration of VMs to the physical servers based on the performance requirements mainly makespan is needed. If a particular VM does not utilizes all the reserved resources, then this VM can be logically resized. This enables us to consolidate VMs to the minimum number of physical machines. Thus the number of active physical servers can be minimized, which in turn reduces the power consumption and reduction in the total energy consumption of the datacenter.

3.2.1.1 Bin packing method

In this proposed work, we adopted a bin packing based approach for VM creation. Here the servers or Physical Machines (PM) in the datacenter are considered as bins. The VMs with user-specified requirements requested by the customers are the objects which are going to be filled in the bins. The algorithm aims to minimize the number of PMs required to place the requested VMs. At the same time, it aims to reduce the makespan of submitted jobs.

The VM assignment problem in the bin-packing method can be defined as follows. Suppose each physical machine PM_i consists of *j* different types of computing resources (R_j) and there are k type of VMs defined by the provider denoted by V_k (where $k = 1, 2, 3, \ldots\ldots, k$). Each physical server PM_i is able to accommodate any kind of virtual machine (V_{kj}) with any type of resource without exceeding its processing capacity.

Let y_i be the number of physical servers, then our aim is to

$$\text{Minimize } z\,(y) = \sum_{i=1}^{n} y_i \qquad (3.1)$$

Subject to the following constraints

1. $\sum_{i=1}^{n} X_{ij} = \begin{cases} 0 \\ 1 \end{cases} \quad \forall i \in \{1, 2, \ldots., N\}$: where X_{ij} = 1 means VM assigned to a physical server and X_{ij} = 0 means it is not assigned to a physical server.
2. $\sum_{i=1}^{n} Load_{ij}\ (t).X_{ij} \leq Load_{max}$: which limits the load to a PM to the maximum predefined $Load_{max}$ for a particular PM at time t.

3. $y_i = \begin{cases} 0 \\ 1 \end{cases} \quad \forall i \in \{1, 2, \dots, M\}$:where 0 means PM$_i$ is not assigned with any VM and 1 if any VM is mapped to it.

3.2.1.2 Best-Fit job placement

In the initial phase jobs are submitted to the using best-fit method. This is to ensure makespan minimization. These jobs are directed to the cloud through the cloud broker. The pseudo code for this best-fit approach is given in figure 3.3.

Algorithm: Best-Fit Job Placement

```
1. Input: JobQueue, VMList
2. Output: Job allocation to VMs in VMList
3. Best_fit_Job()
4.      Sort VMs based on processing power in ascending order
5.      Sort Jobs in ascending order based on MIPS required
6.      AssignedJobList ← NULL
7.      Set VMStatus = 0                //All VMs are job free
8.      Set JobStatus = 0               //No Jobs are allocated to VMs
9.      for each Job i in JobQueue do
10.             for each VM j in VMList do
11.             If Power of VM[j] ≥ Job[i] && VMStatus = 0 then
12.                     Assign Job[i] to VM[j]
13.                     Set VMStatus = 1
14.                     Set JobStatus = 1
15.             Else
16.                     Append Job[i] to UnAssignedJobList[]
17.             End if
18.             End for
19.     End for
```

Fig. 3.3 Best-Fit job placement

Algorithm: Remaining-Fit VM Placement

```
1. Input: VMList, PMList
2. Output: Mapping of VMs to PMs
3. Worst_fit_VM()
4.      Sort VMList in ascending order.
5.      Sort PMs in the PMList in descending order.
6.      UsedPMList = 0, UnusedPMList = 0
7.      currentPMstatus = 0 for all PMs          // PM not yet allocated.
8.      start ← start_VM and last _ last_VM in the VMList
9.      for each PM j in PMList do
10.         for each VM i in VMList do
11.         if UnusedResource(PM_j) ≥ ResourceNeed(VM_i) then
12.            if PMstatus_j == 0 then
13.                 Add VM_i to PM_j
14.                 PM_j ← PM_j - VM_i
15.                   else
16.                   Add VM_i to PM_j
17.                   PM_j ← PM_j - VM_i
18.                   Until last VM
19.                 End if
20.                   else
21.                   Set PMstatus_j =1  // PM_j in PMList is allocated
22.                   Add PMj to usedPMList
23.                   Start_VM = next VM in VMList (VM_{i+1})
24.                 Until last PM in PMList
25.         End if
26.         End for
27.     End for
```

Fig. 3.4 Remaining-Fit VM placement

Based on the processing power requirement of each job, it is sorted in ascending order. Currently available VMs are also sorted in a list based on their processing capacity. After this, the cloud broker places the jobs from the job queue to these available VMs. If a particular job *i* is assigned to the VM_j, then the algorithm changes its status to 1 for both job *i* and VM*j*.

3.2.1.3 Remaining-Fit VM placement

Next in the second phase, the optimization of VM allocation to the physical hosts is carried out worst-fit method.

Here the Physical Machines are sorted in decreasing order of utilization and VMs are sorted based on Million Instruction Per Second (MIPS). Then, the algorithm finds the first PM the list of sorted PMs and places the first VM from the sorted VMList to this selected PM. This process is continued till every VM_i in the list are mapped to PM_j. The procedure for the Worst-fit method for VM placement is given in figure 3.4. Thus VMs are placed to its maximum capacity; hence the number of PMs required for hosting VMs can be minimized. This reduction of active physical servers will reduce the power consumption of the datacenter.

3.3 Experimental Setup and Results

We have evaluated the effectiveness of bin packing method using CloudSim [129]. It is a powerful simulation tool to test scheduling and load balancing mechanisms.

3.3.1 Simulation environment

We have deployed 10 physical machines. These 10 physical servers together can accommodate up to 100 virtual machines. These PMs with memory size varies between 1 to 3 GB. The memory capacities of the VMs are configured within the ranges from 100 MB to 1 GB. The power consumption of each PM is measured using the built-in power datacenter in CloudSim. The storage requirement of each submitted job can vary between 100 to 800 MB randomly. The storage capacity of each physical server is fixed up to 1TB.

3.3.2 Evaluation parameters

The performance of the method is measured based on the number of physical machines used, power utilization and storage comparison. The method is experimented and all these parameters are measured for Best-Best, Best-Remaining, Worst-Worst and Worst-Best strategies. All these algorithm combinations are simulated in the same environment.

3.3.2.1 Number of PMs used

When the active number of PMs increases, that will also increase the power consumption. The comparative performance of the above algorithms for the active number of PMs is shown in figure 3.5. It shows that the significant improvement in power consumption using the proposed method.

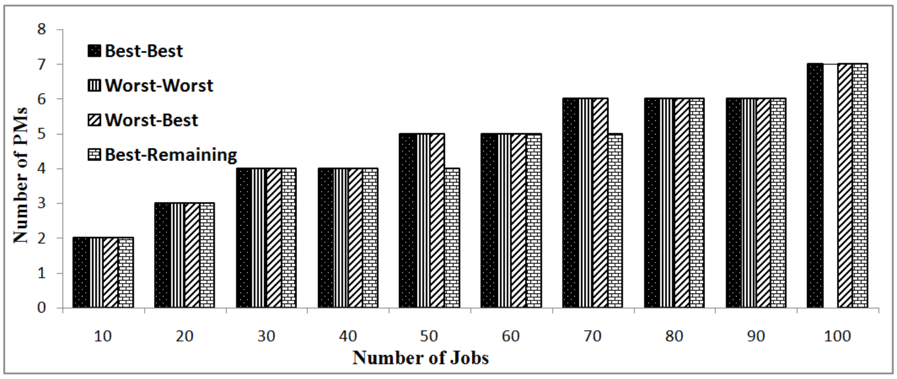


Fig. 3.5 Comparison – Number of PMs used

The algorithm searches among all the underutilized PMs to find the appropriate server for the placement of VMs with user-specified requirements. The proposed method helps in the efficient use of active servers. Thus it avoids usage of extra PMs to accommodate virtual machines. Reduction in the active PMs, in turn, reduces the power consumption that reduces the computation cost.

3.3.2.2 Storage space

We have measured the storage allocation efficiency of the proposed method. The results are compared and it is plotted in figure 3.6. From the figure, we can see that the proposed method uses less percentage of space compared to other strategies. The proposed method effectively uses the available storage space in the active PMs.

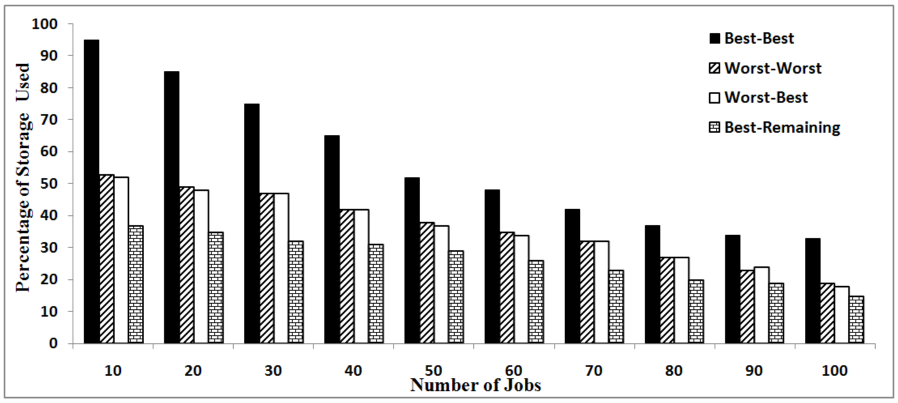


Fig. 3.6 Storage space utilization

3.3.2.3 Power utilization

The cloud providers always looking for reduced power utilization. Extensive simulations were carried out and the power utilization of the proposed method is compared for different number of jobs with different number of PMs. The results are shown in figure 3.7. From the above figure, we can conclude that the power consumption analysis is comparatively promising. This is because the proposed method uses less number of PMs for placing user-requested VMs.

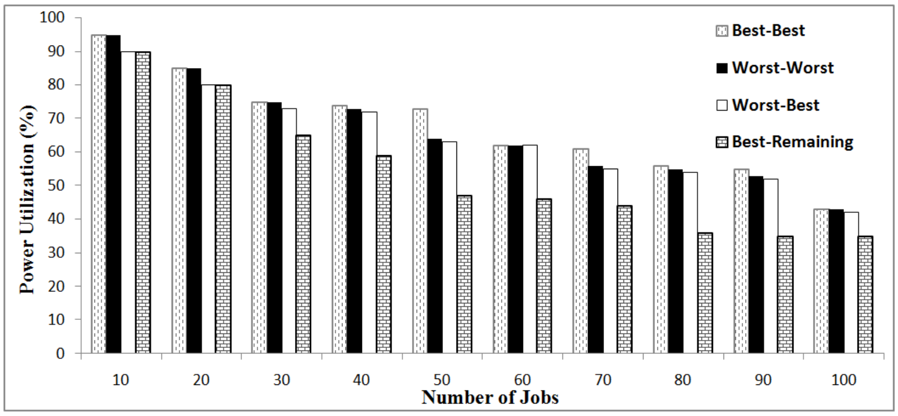


Fig. 3.7 Power utilization

3.3.2.4 Makespan

From figure 3.8, it is clear that the proposed Best-Remaining fit strategy reduces the makespan of jobs. Thus effective handling of makespan by the proposed bin-packing based technique improved the QoS in terms of makespan.

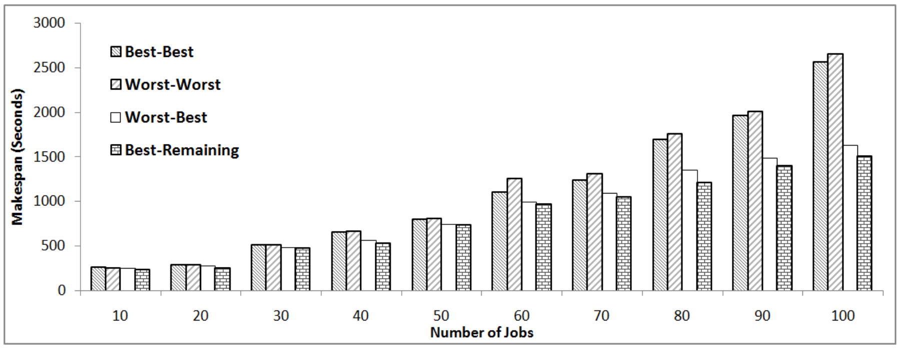


Fig. 3.8 Makespan

3.4 Benefits of Bin packing

The proposed Bin packing method is compared with its other variants like First Fit Decreasing (FFD) [193] and Max-Min [194] algorithms. The results shown in figure 3.9 indicates that the proposed method gives significant performance improvement in terms of makespan.

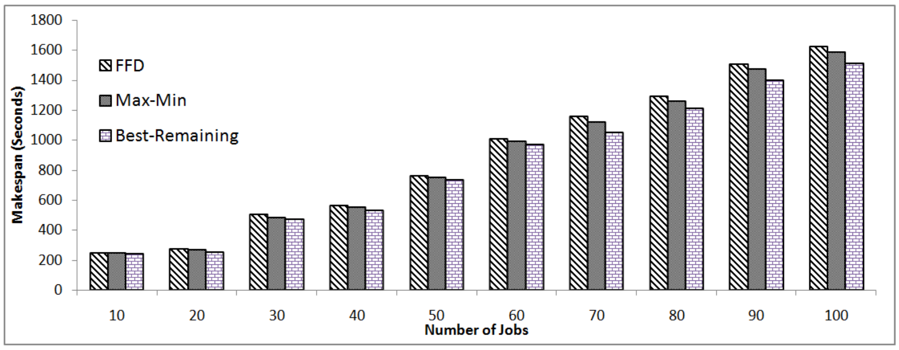


Fig. 3.9 Comparison with FFD and Max-Min algorithms

3.5 Summary

In cloud computing, the effective and efficient use of cloud resources is crucial for the service provider's revenue. Usually, one of the QoS parameters requested by the customer is makespan. Also, in order to harness the green energy concept, the importance of improved energy efficiency mechanisms is to be considered. This chapter proposed a Best-fit - Remaining-fit strategy that efficiently handles makespan, thus improving QoS. It also places the virtual machines to a minimum number of active physical servers. By the simulated study, we have shown the effectiveness of Best-fit – Remaining-fit technique in handling makespan.

CHAPTER 4
ENHANCED LOAD BALANCING FOR VM MIGRATIONS

Contents

4.1 Introduction

The load balancing method avoids under and heavy loaded conditions in the datacenters. When some resources are overloaded with several number of tasks, these tasks are to be migrated to the under loaded resources of the same datacenter in order to maintain QoS. Frequent VM migrations also affect the performance of the cloud ecosystem. Nature inspired algorithms are efficient in solving this kind of dynamic problems. In this chapter, we proposed an enhanced bee colony algorithm for efficient and effective load balancing in the cloud environment. The honey bees foraging behaviour is used to balance load across virtual machines. The tasks removed from overloaded VMs are treated as honeybees and underloaded VMs are the food sources. The method also tries to minimize makespan as well as number of VM migrations. The algorithm also reduced the imbalance in the cloud eco system. The experimental result shows that there is significant improvement in the QoS delivered to the customers.

4.1.1 How migrations affect makespan

In order to ensure QoS efficient load balancing among nodes are required in the distributed cloud environment. An efficient load balancing mechanism tries to speed up the execution time of user-requested applications. It also reduces system imbalance and gives a fair response time to the users. VM migrations are to be carried out for load balancing.

When migration is happening, the currently execution VM stops and some time is required to restart at a new location. This delay causes a potential impact on the makespan. So migration reduction is an important factor to maintain QoS in the cloud. In order to limit migrations, a better load balancing mechanism is needed.

The better load balancing will result in reduce response and migration time. The improvement in the above factors will ensure good QoS to the customers thereby less Service Level Agreement (SLA) violations.

Static load balancing algorithms will work only when there is a small variation in the workload. Cloud scheduling and load balancing problems are considered as NP-hard problems. The dynamic nature of the cloud computing environment needs dynamic algorithms for efficient and effective scheduling and load balancing among computing nodes.

4.1.2 Artificial Bee Colony algorithm

The Bee Colony algorithm is a meta-heuristic swarm intelligence algorithm [130] to solve numerical function optimization problems. It mimics the foraging behavior of honey bees. It has advantages such as memory, multi character, local search, and solution improvement mechanism, so it is an excellent solution for optimization problems [143, 144, 145].

The Bee Colony consists of three groups of artificial bees: employed foragers, onlookers, and scouts. The employed bees comprise the first half of the colony whereas the second half consists of the onlookers. The employed bees are linked to particular food sources. In other words, the number of employed bees is equal to the number of food sources for the hive. The onlookers observe the dance of the employed bees within the hive, to select a food source, whereas scouts search randomly for new food sources.

The search cycle of Artificial Bee Colony consists of three rules:

- Sending the employed bees to a food source and evaluating the nectar quality

- Onlookers choosing the food sources after obtaining information from employed bees and calculating the nectar quality
- Determining the scout bees and sending them onto possible food sources

The positions of the food sources are randomly selected by the bees at the initialization stage and their nectar qualities are measured. The employed bees then share the nectar information of the sources with the bees waiting at the dance area within the hive. After sharing this information, every employed bee returns to the food source visited during the previous cycle, since the position of the food source had been memorized and then selects another food source using its visual information in the neighbourhood of the present one.

Algorithm: Artificial Bee Colony Algorithm

```
1. Initialize the Bee Colony and problem parameters
2. Initialize the Food Source Memory (FSM)
3.    Repeat
4.        Send the employed bees to the food sources.
5.        Send the onlookers to select a food source.
6.        Send the scouts to search for possible new food.
7.        Memorize the best food source.
8.     Until termination criterion is met
9. End
```

Fig. 4.1 Artificial Bee Colony algorithm

At the last stage, an onlooker uses the information obtained from the employed bees at the dance area to select a food source. The probability for the food sources to be selected increases with an increase in its nectar quality. Therefore, the employed bee with information on a food source with the highest nectar quality recruits the onlookers to that source. It subsequently chooses another food source in the neighbourhood of the one currently in her memory based on visual information (i.e. comparison of food source positions). A new food source is randomly generated by a scout bee to replace the one abandoned by the onlooker bees. This search process is represented shown in figure 4.1 [146].

4.2 Related Works

Efficient scheduling and load balancing ensures better QoS to the customers and thereby reduces the number of SLA violations. This section reviews some of the load balancing algorithms.

Modified throttled algorithm based load balancing is presented in [131]. While considering both the availability of VMs for a given request and uniform load sharing among the VMs for number of requests served, it is an efficient approach to handle the load at servers. It has an improved response time, compared to existing Round-Robin and throttled algorithms, but it suffers from frequent migrations.

In [132], a load balancing approach was discussed, which manages load at server by considering the current status of all available VMs for assigning the incoming requests. This VM-assign load balancing technique mainly considers efficient utilization of the resources and VMs. By simulation, they proved that their algorithm distributes the load optimally and hence avoids under/over utilization of VMs. The comparison of this algorithm with an active-VM load balance algorithm shows that their algorithm solves the problem of inefficient utilization of the VMs.

Response time based load balancing is presented in [133]. In order to decide the allocation of new incoming requests, the proposed model considers current responses and its variations. The algorithm eliminates the need for unnecessary communication of the Load Balancer. This model only considers response time which is easily available with the Load Balancer as each request and response passes through the Load Balancer, hence eliminates the need of collecting additional data from any other source thereby over utilizing the communication bandwidth.

In [134] a load balancing technique for cloud datacenter, Central Load Balancer (CLB) was proposed, which tried to avoid the situation of overloading and under loading of virtual machines. Based on priority and states, the Central Load Balancer manages load distribution among various VMs. CLB efficiently shares the load of user requests among various virtual machines.

Ant colony based load balancing in cloud computing was proposed in [135]. It works based on the deposition of pheromone. A node with minimum load is attracted by most of the ants. So maximum deposition of pheromone occurs at that node and performance is improved.

Cloud Light Weight (CLW) for balancing the cloud computing environment workload is presented in [136]. It uses two algorithms namely, receiver-initiated and sender-initiated approaches. VM Attribute Set is used to assure the QoS. CLW uses application migration (as the main solution) instead of using VM migration techniques to assure minimum migration time.

A resource weight based algorithm called Resource Intensity Aware Load balancing (RIAL) is proposed in paper [73]. In this method, VMs are migrated from over-loaded Physical Machines (PM) to lightly loaded PMs. Based on resource intensity the resource weight is determined. A higher-intensive resource is assigned a higher weight and vice versa in each PM. The algorithm achieves lower-cost and faster convergence to the load balanced state, and minimizes the probability of future load imbalance, by considering the weights when selecting VMs to migrate out and selecting destination PMs.

A cloud partitioning based load balancing model for the public cloud was proposed in [137]. This algorithm applies game theory to load balancing strategy in order to improve efficiency. Here a switching mechanism is used to choose different strategies for different situations.

Time and cost based performance analysis of different algorithms in cloud computing were given in [138]. A load balancing mechanism based on artificial bee colony algorithm was proposed in [139] but it suffers from frequent migrations. It optimizes cloud throughput by mimicking the behavior of honey bees. Since the bee colony algorithm arranges only a little link between requests in the same server queue, then maximization of the system throughput is suboptimal. Here, the increasing request does not lead to an increase in system throughput in certain servers.

An active clustering based load balancing technique is presented in paper [140]. It groups similar nodes together and works on these groups and produces better performance with high utilization of resources.

Weighted Signature based Load Balancing (WSLB), a new VM level load balancing algorithm is presented in [141]. This algorithm finds the load assignment factor for each host in a datacenter and map the VMs according to that factor. Estimated finish time [142] based load balancing considers the current load of virtual machines in a datacenter and the estimation of processing finish time of a task before any allocation. This algorithm improves performance, availability and maximizes the use of virtual machines in their datacenters. In order to avoid a probable blocking of tasks in the queue, it permanently controls the current load on the virtual machines and the characteristics of tasks during processing and allocation.

The authors in [148] proposed a heuristic based scheme for load balancing in the large cloud data centers based on duplicating jobs and sending replicas to different servers. They showed that this mechanism can significantly reduce the queuing time, even with a small number of replicas and in particular in high workloads. Determining the right parameter configuration for this method (the number of replicas, the server job selection policy) is highly dependent on the system condition, comprising the scale of the system, load pattern, job processing time, and inter-server delays. As different systems may be subject to different conditions, there is no single parameter configuration that is optimal to all systems. So in order to deploy the scheme, the system manager should conduct a simulation-based study to determine the right settings for the specific system. Cloud is a dynamic environment, so the conditions may change. Therefore, system performance should be constantly monitored in order to determine whether any of the parameter values should be modified.

A dynamic load-balanced scheduling (DLBS) based on heuristic algorithms approach to maximize the network throughput through dynamically balancing data flows is developed in [149]. In this method, the data flow is balanced time slot by time slot. The simulation result shows that the algorithm works better when data flow is high.

There are several methods proposed for load balancing in the cloud such as collaborative agents for distributed problem solving [150], CLB load balancing architecture and algorithm [151], Temporal task scheduling with heuristics [152], QoS based methods [153], and concave pricing [154].

4.3 Proposed Method

When the workload increases, load balancing is an important task in resource management to ensure quality of service. An optimal task scheduling algorithm is needed for the load balancing problems as well as users' expectations in QoS. The load balancing algorithm called Interaction ABC (IABC) [239] is based on bee colony to schedule the tasks to virtual machines (VMs), but number of migration is very high. The paper [147] also tried bee colony algorithm for load balancing in the cloud, but still, frequent migration is a problem.

So in our proposed method, we have considered completion time of tasks and number of task migrations along with system imbalance during computation.

4.3.1 Architecture

The architecture for our proposed load balancing method is shown in figure 4.2. The details of each component are given below.

Cloud Information Service (CIS): It is the repository that contains all the resources available in the cloud environment. It can be considered as a registry of datacenters. Whenever a datacenter is created it has to register to the CIS and update details.

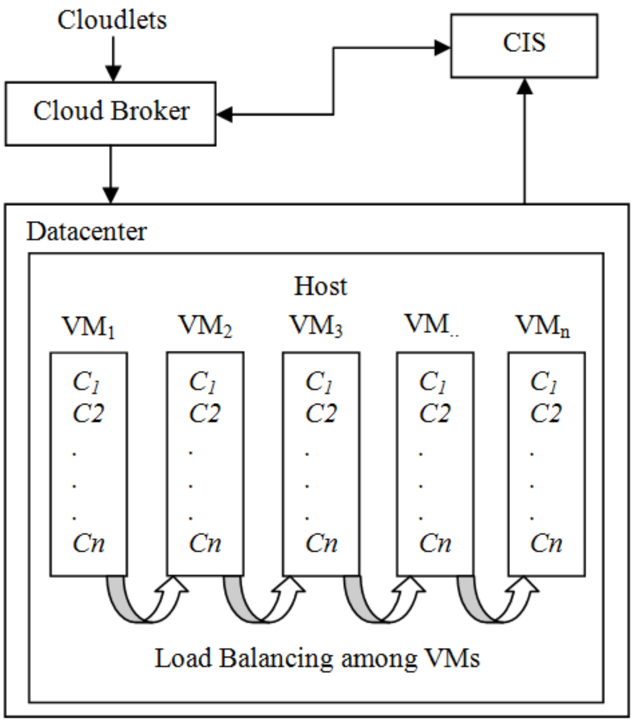


Fig. 4.2 Load balancing architecture

Datacenter: Here we have considered Datacenters with heterogeneous resources. A datacenter consists of several hosts. Each host can contain many processing elements (PEs) with RAM and bandwidth characteristics. Based on the user requirement, the hosts are virtualized into different number of VMs. VMs may also have heterogeneous nature as like hosts.

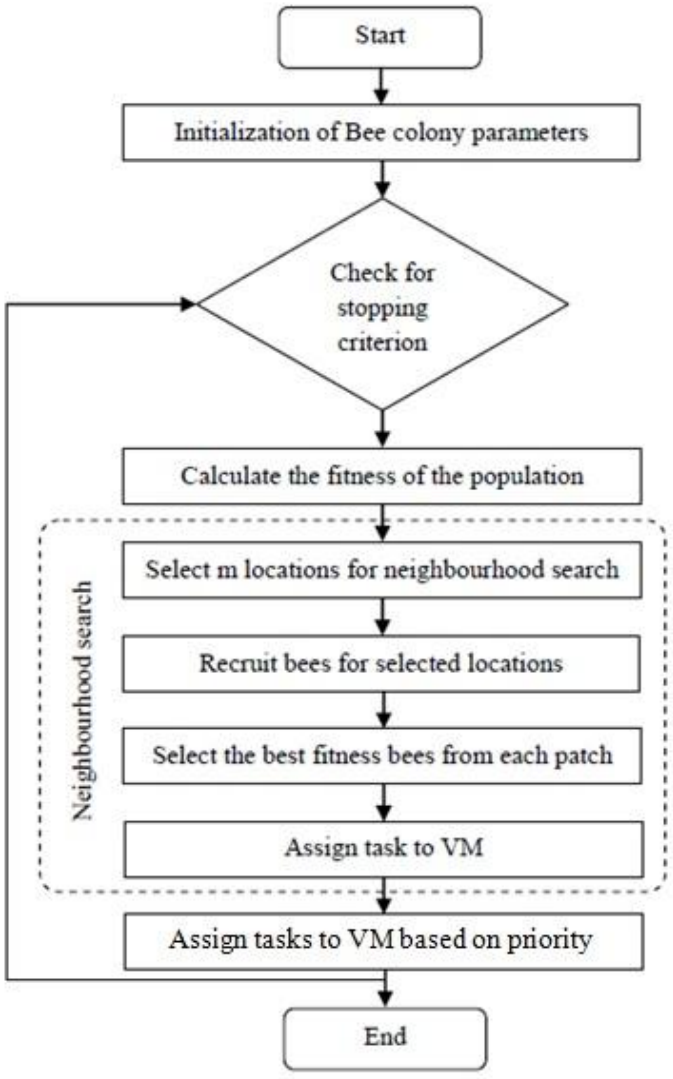

Start
Initialization of Bee colony parameters
Check for stopping criterion
Calculate the fitness of the population
Neighbourhood search
Select m locations for neighbourhood search
Recruit bees for selected locations
Select the best fitness bees from each patch
Assign task to VM
Assign tasks to VM based on priority
End

Fig. 4.3 Enhanced Load balancing using bee colony algorithm

The role of CIS is to collect information about all the resources in the datacenters. This information used for the submission of tasks to the physical hosts.

The enhanced bee colony algorithm is given in figure 4.3.

4.3.2 Steps for cloud load balancing

The basic steps used for cloud load balancing are given in the figure 4.4.

1. Start
2. Find load of each VMs and group VMs as over-loaded or under loaded.
3. Find the supply of under loaded VMs and demand of overloaded VMs.
4. Sort the overloaded and under loaded VM sets
5. Sort the tasks in overloaded VMs based on priority.
6. For each task in each overloaded VM find a suitable under loaded VM.
7. Update the overloaded and under loaded VM sets and go to step 2.
8. Stop

Fig. 4.4 Steps for cloud load balancing

4.3.3 Parameter mapping

The proposed method used the foraging behaviour of honeybees for effective load balancing across VMs in the datacenters and reschedules the tasks to the under loaded VMs. For the implementation of bee algorithm in the cloud, the characteristics of honeybees are to be mapped into the cloud environment. The mapping of bee colony parameters with the cloud environment is shown in table 1.

Table 4.1: Mapping of Bee colony parameters with Cloud environment

Honey Bee Hive	**Cloud Environment**
Honey bee	Task (Cloudlet)
Food source	VM
Honey bee foraging a food source	Loading of a task to a VM
Honey bee getting depleted at a food source	VM in overloaded condition
Foraging bee finding a new food source	Removed task will be rescheduling to an under loaded VM having highest capacity

4.3.4 Load balancing

In this proposed method the tasks are considered as honeybees. When honey bee forage for food source, then the cloudlet will be assigned in VM for execution. Since the processing capacity varies for different VMs, sometimes VMs may be overloaded and others will be underloaded. In these circumstances, an efficient load balancing mechanism is needed. When a particular VM is overloaded then some tasks need to be migrated away and have to assign it to an under loaded location. In this case, the task to be migrated is chosen based on priority. In the proposed method tasks with the lowest priority will be selected as a candidate for the migration. This procedure is similar as honey is exhausted in nectar and bees are ready to take off from the food source.

The fitness function used is based on the task length and processing capacity of a VM. Equation (4.1) gives fitness value of VM *j* for a task *i*.

$$Fit_{ij} = \frac{\sum_{i=1}^{i=n} Task\ Length_i}{Processing\ \ Capacity\ \ of\ VM_j} \tag{4.1}$$

The tasks are assigned to a particular VM is based on this fitness value.

The proposed method works in four different steps as given below.

1. VM Current Load Calculation
2. Load Balancing & Scheduling Decision
3. VM Grouping
4. Task Scheduling

VM Current Load Calculation: The current load on a VM is measured based on the ratio between total lengths of the tasks submitted to that VM to the processing rate of that VM at a particular instance. Suppose N is the total number tasks assigned to a VM and Len is the length of single tasks and MIPS is the Million Instruction Per Second rate of that VM, then using the equation (4.2) the current load can be calculated.

$$Load_{VM} = \frac{N * Len}{MIPS} \tag{4.2}$$

Then total load on a datacenter is the sum of load on each VMs. The equation for total load a datacenter $Load_{DC}$ is given by the equation (4.3).

$$Load_{DC} = \sum_{i=1}^{m} Load_{VM_i} \tag{4.3}$$

The processing capacity of VM can be calculated using the equation (4.4) as given below.

$$Capacity_{VM} = PE_{num} * PE_{mips} + VM_{bw} \tag{4.4}$$

Here PE_{num} is the number of processing elements in a particular VM, PE_{mips} is the processing power of PE in MIPS rate and VM_{bw} is the band width associated for a VM.

A datacenter may have several VMS. So the total capacity of the entire datacenter can be calculated from using the equation (4.5),

$$Capacity_{DC} = \sum_{vm=1}^{n} Capacity_{VM} \tag{4.5}$$

Then the proposed algorithm computes the processing time of each task using equation (4.6).

$$PT = \frac{Current\ \ Load}{Capacity} \tag{4.6}$$

Then the processing time required for datacenter to complete all the tasks in it can be calculated by the equation (4.7) given below,

$$PT_{DC} = \frac{Load\ _{DC}}{Capacity\ _{DC}} \tag{4.7}$$

Then the Standard Deviation (SD) is a good measure of deviations. The proposed method uses SD for measuring the deviations in the workload on each VM. If there is m VMs, then Equation (4.8) gives the SD of loads.

$$SD = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_i - PT)^2} \tag{4.8}$$

Then the load balancing decision is done based on the value of SD.

In this proposed method, bee colony algorithm is modified to find optimal solution quickly. This algorithm quickly converges into an optimal solution. The algorithm also tries to minimize the number of task migrations. It also considers users' priority while scheduling the tasks.

Algorithm: Steps for Enhanced Bee Colony

```
1. Start
2. For each task do
3.      Calculate the load on VM and decide whether to do load balancing
        or not
4.      Group the VMs based on load as overloaded or under loaded based
        on fitness value.
5.      Find the supply of under loaded VMs and demand of overloaded
        VMs.
6.      Sort the overloaded and under loaded VM sets
7.      Sort the tasks in overloaded VMs based on priority.
8.      Find the capacity of VMs in the under loaded set.
        For each lower priority task in the overloaded VM find a suitable
        under loaded VM based on capacity.
9.      Update the overloaded and under loaded VM sets
10. End of step 2.
11. Stop
```

Fig. 4.5 Enhanced Bee colony based load balancing algorithm

Load Balancing & Scheduling Decision: In this phase, load balancing and rescheduling of tasks are decided. This decision depends on the SD value calculated using equation (4.8). In order to maintain system stability, the load balancing and scheduling decision will take only when the capacity of the datacenter is greater than the current load. Otherwise, it will create an imbalance in the datacenter. For finding the load threshold value is set (value lies in 0-1) based on the SD calculated. The systems compare this value with the calculated SD measure. The load balancing and scheduling is done only if the calculated SD is greater than the threshold. This will improve system stability by minimizing the number of migrations.

VM Grouping: In order to increase the efficiency VMs are grouped into two groups: overloaded VMs and under loaded VMs. This will reduce the time required to find optimal VM for task migration. The overloaded VMs are the candidates for migration. In the proposed method these removed tasks are considered as honeybees and the under loaded VMs are their food sources. The VMs are grouped according to the SD and threshold value already calculated based on the load.

Task Scheduling: Before initiating load balancing the system have to find the demand to each overloaded VMs and supply to the under loaded VMs. Here the VMs are sorted based on the capacity in ascending order. The task migration is performed only when demand meets the supply. From the under loaded VM set, the proposed method selects a VM which has the highest capacity as target VM. The method selects the task with the lowest priority from an overloaded VM and it is rescheduled to an under loaded VM with maximum capacity.

Supply to a particular VM is the difference between its capacity and current load and it can be calculated using equation (4.9),

$$Supply_{VM} = Capacity - Load \tag{4.9}$$

Then the demand of a VM is calculated using the equation (4.10)

$$Demand_{VM} = Load - Capacity \tag{4.10}$$

On submission of each task into the cloud, the VM will measure the current load status and calculates SD. If the SD of loads is greater than the threshold then load balancing process is initiated. During this load balancing process, VMs are classified into under loaded and overloaded VM sets. Then the submitted tasks are rescheduled to the VM having the highest capacity.

4.4 Experimental Results

The proposed method is tested in the simulated cloud environment using CloudSim. VMs with different specifications are deployed. User requests are submitted to this heterogeneous environment. We have measured the number of VM migrations, makespan and degree of imbalance are measured and compared these parameters with existing methods.

4.4.1 Makespan

Here we have compared makespan of the enhanced bee colony algorithm with bee colony method. Also, it is compared with RR [166] and Max-Min [194] algorithms. The migration time i.e., overall task completion time is graphically represented in figure 4.6. The results indicate that the proposed method reduced the makespan than bee colony algorithm and other existing methods. From the results, it is clear that makespan can be reduced into a significant amount using load balancing algorithm. Makespan is a good measure of QoS provided by the service provider. So the proposed method improves the QoS.

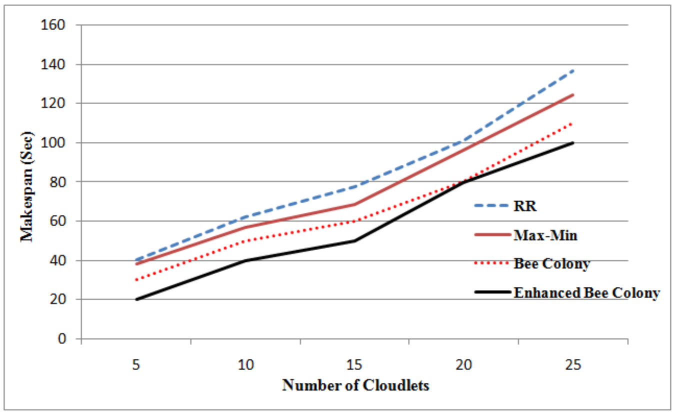


Fig. 4.6 Comparison of makespan

4.4.2 Number of migrations

If the tasks are continuously shifting from one assigned location to another, it adversely affects the performance of the system. So the number of migrations is a performance indicator to measure the performance of a service provider. In the proposed method the algorithm considers the priority of tasks when migration is needed. If lower the priority of a task, there is a higher chance for migration from the assigned queue. It is to ensure the higher priority tasks are less affected. The results are represented in figure 4.7. From the result, we can observe that the enhanced version outperforms bee colony algorithm in most cases.

The above test results show how the proposed method reduces the makespan as well as the number of task migrations. Thus it helps to improve the performance of the cloud service provider.

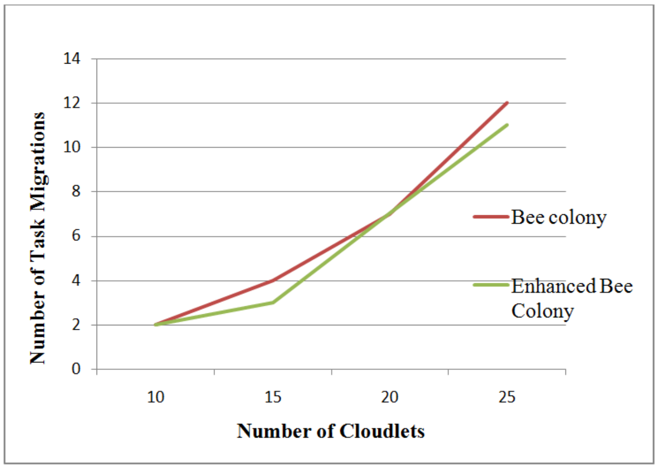


Fig. 4.7 Number of task migrations

4.4.3 Degree of imbalance

Another performance parameter is the degree of imbalance. This is to measure system stability due to migrations. Table 4.2 and figure 4.8 represents the degree of imbalance before and after applying the algorithm. From the results, it is clear that the proposed method reduced the imbalance due to migrations since this method employs less number of migrations.

Table 4.2: Degree of Imbalance

Number of Cloudlets	Before	After
10	1.000	0.714
15	0.509	0.664
20	0.840	0.268
25	0.667	0.212

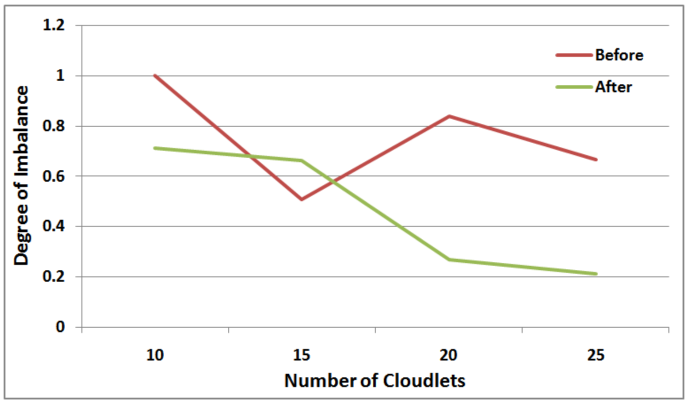


Fig. 4.8 Degree of Imbalance before and after applying the algorithm

4.5 Summary

In this chapter, we have proposed and experimented a bee colony algorithm for efficient load balancing in the cloud environment. In this method, we have used the power of swarm intelligence algorithm to remove the tasks from overloaded resources and migrated these removed tasks to the most appropriate underutilized or under loaded resources. This migration policy also considers the priority of the tasks in the waiting queue. The tasks with the least priority are selected as candidates for migration. Hence, no tasks are needed to wait a long time to get processed and reduced makespan migrations. The experimental results show a lower number of migrations with a reduced imbalance for the proposed approach.

CHAPTER 5
LOAD BALANCING FOR IMPROVING ENERGY EFFICIENCY

Contents

5.1 Introduction

Cloud has several advantages such as availability, scalability, and reliability, but some performance parameters such as energy consumption, load balancing, response time, resource allocation time, etc., need further attention. The cloud consists of several huge datacenters, each with heterogeneous physical machines. When the number, diversity, and size of the datacenters are increasing, optimal resource identification and allocation needs a high resource discovery time.

This chapter proposes an energy-aware clustered load balancing method. In this method, first, heterogeneous resources are grouped into different clusters by using a partitioning based clustering algorithm. Clustering reduces the number of resources needed to be searched and therefore, the total searching time required for resource discovery and allocation can be reduced. Since the search process is carried out only on a particular cluster, the searching time will be reduced. In the next phase of the method, an energy-aware best-fit VM allocation is carried out based on the weight value of the resource. This weight value depends on its memory, storage and processing capacity of the resource. Then the corresponding VM cluster is found out using this weight value. If suitable resources are available in that cluster, then allocate it. Else, searching progresses towards second portion of that cluster for the resource availability. If the VM is unable to allocate in that cluster, then the method checks in other clusters. Finally, a best-fit allocation strategy is used for allocating processes to the VMs. The best-fit algorithm helps in efficient VM placement for optimal space utilization. We have also implemented Ant Colony Optimization (ACO) based clustering method and compared it with energy-aware clustering method.

5.1.1 Energy Management

Load balancing methods are good for handling huge requests efficiently and for placing the VMs in particular PMs [169]. The load balancing mechanisms consume a large amount of power during VM

Migration, task execution, etc. This is due to the use of all available PMs in the datacenter to maintain quality or due to the absence of good power management policies. If an energy-aware load balancing mechanism is applied to the clusters to balance the load among PMs, that will improve power consumption. i.e., it will help to identify and switch off the idle physical machines. The proposed method also aims at achieving high user satisfaction by minimizing the response time, improvement in resource utilization through an even and fair allocation of cloud resources with reduced energy usage.

5.2 Related Works

CIVSched [157] is communication-aware inter-VM scheduling focused to reduce network latency between co-located VMs. In this method, VM Monitor (VMM) monitors traffic and schedule the processes. The scheduling process is done by the cooperation of VMM and the guest local operating system, but the power consumption is not considered in this system.

The green cloud computing [156] method mainly aims to reduce the carbon emission and energy consumption in the distributed cloud datacenters having different sources of energy and carbon footprint rates. Here the rate of carbon footprints at datacenters is used for VM migration and allocation. The datacenter power usage details are given in [167] indicate that a large amount of power is wasted due to inefficient resource management mechanisms. Learning automata [159] based method is one of the approaches to improve resource utilization with energy consideration.

Context switching [155, 164] will reduce the speed of the system as well as it increases the power consumption during packet transfer. This is because the state must be saved and restored, even if much of the restored state is not used before the next context switch. Here a context cache is introduced, which is used to bind variable names to individual registers. It allows context switches to be very inexpensive because registers are only loaded and saved as needed. From the analysis, the context cache contains more live data than a multithreaded register file and supports more tasks without spilling to the memory.

The advanced version of Minimum Laxity First (MLF) called optimized MLF (OMLF) [160] try to reduce context switches. This dynamic algorithm is proposed to overcome the problems in MLF algorithm and makes it more suitable for spacecraft avionics systems. The OMLF is tested using mathematical modeling and simulation tools. The results are promising and it has fewer context switches than the traditional MLF. To make it more suitable for dynamic cloud scheduling other cloud parameters are to be considered. In [161], First Fit Decreasing (FFD) based energy-aware scheduling and workload consolidation algorithms are presented. Both these methods check the problem of grouping heterogeneous workloads. They try to accommodate all the VMs to the minimum number of PMs and then turn off unused physical servers to reduce energy consumption. Even though the energy considerations are integrated into the algorithm, it works in three phases that cause some delay in response time.

Another method based on clustering is the Multi Queue Scheduling (MQS) algorithm that is proposed to reduce the cost of reservation and on-demand charges using a global scheduler [163]. In this method, the global scheduler shares the resources at the maximum level. Here the jobs are clustered on the basis of burst time. It also overcomes the fragmentation problems in classical scheduling methods such as First Come First Serve, Shortest Job First, EASY, Combinational Backfill and Improved backfill. It is suitable for continuous workflows. The enhanced version of MQS algorithm [162] uses a fuzzy logic concept. The fuzzy logic improves efficiency and it comes up with the best option for shifting the load from one location to another. In this method also the fuzzy logic mechanism reduces overall overheads of the live migration techniques but this method is limited to continuous workflows.

Some algorithms groups incoming jobs in the queue to increase efficiency. The tri queue scheduling [165] is one such method in which based on the processor requirement of the job, the queue is grouped into small, medium and long jobs. It is a dynamic quantum time based round robin scheduling mechanism. Even though it is a job grouping mechanism to improve performance, the energy and cost factor does not consider for the scheduling tasks. An enhanced weighted round robin (EWRR) [166] method proposed an energy-efficient job scheduling. It is an advanced form of weighted round robin scheduler that considers VMs reuse and live VM migration. Moreover, this algorithm is integrated with DVFS algorithm in CPU utilization model to specify the required frequency for each task depending on the task complexity and deadline and experiments were conducted with a very little number of PMs.

Table 5.1 Notations used

Notation	**Definition**
$W_{PM(i)}$	Weight value of a PM
$M_{PM(i)}$	Memory of *i*th PM
$M_{Max\ PM(i)}$	Maximum memory of a PM
$S_{PM(i)}$	Total storage of a PM
$S_{MaxPM(i)}$	Maximum storage capacity of a PM
$P_{speed\ PM(i)}$	Processor speed of *i*th PM
$P_{MaxPM(i)}$	Maximum allowed processor speed
$W_{VM(i)}$	Weight value of a VM
$M_{VM(i)}$	Required Memory of *a* VM
$M_{Max\ VM(i)}$	Maximum allowed memory of a VM
$S_{VM(i)}$	Storage capacity of a VM
$S_{MaxVM(i)}$	Maximum storage allowed for a VM
$P_{VM(i)}$	Processing power of VM
$P_{MaxVM(i)}$	Maximum allowed processor speed of VM
C_i	Cluster *i*
I_{max}	Maximum value of *Weight* for a cluster
I_{min}	Minimum value of *Weight* for a cluster
$W_{P(i)}$	Weight value of a process
$M_{P(i)}$	Required Memory of a process
$M_{Max\ P(i)}$	Maximum allowed memory of a process
$S_{P(i)}$	Required storage capacity of a process
$S_{MaxP(i)}$	Maximum storage allowed for a process
$P_{P(i)}$	Required processing power of a process
$P_{MaxP(i)}$	Maximum allowed processor speed of a process
θ	Pheromone evaporation rate
FP	Foraging pheromone
TP	Trailing pheromone

All the above studies show that energy-aware resource management in the dynamic cloud is a big challenging problem since several business organizations are adopting this platform. There are several other methods proposed for load balancing such as profit and energy based method [158], reservation based [170], task based [171] and an energy-conscious task consolidation heuristics [168].

5.3 Proposed System

We have used a partitioning based clustering algorithm to group PMs in our proposed method. Our proposed clustering method is shown in figure 5.1. It consists of *n* number of users with *n* processes. Each process requires at least one VM to satisfy its requirements. All the physical machines in the datacenter are grouped into different clusters. Depending on the processing capacity the number of PMs in each cluster may be different. The notations used in this chapter are shown in table 5.1.

The proposed method consists of three steps as given below.

1. Clustering of PMs,
2. Energy aware PM migration
3. Process allocation.

The detailed description of the above three steps are explained in 5.3.1 to 5.3.4.

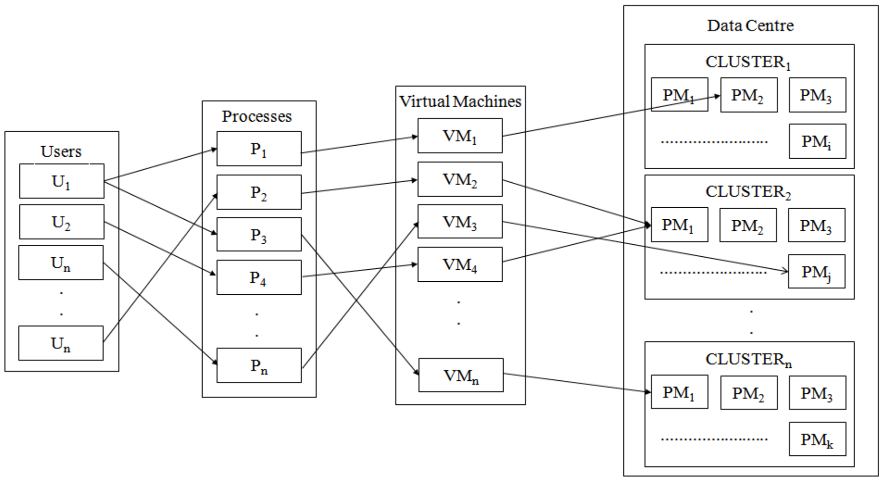


Fig. 5.1 Proposed system architecture

5.3.1 Clustering of physical machines

Clustering needs some criteria to group similar physical machines. Here we have employed a clustering algorithm based on the characteristics of the PMs. The parameters considered are processor speed, memory and storage capacities of PMs. For each PM a weight value $W_{PM(i)}$ is calculated using the equation (5.1) based on these parameters.

$$W_{PM(i)} = M_{PM(i)}/M_{MaxPM\,(i)} + S_{PM\,(i)}/S_{MaxPM(i)} + P_{speed\;PM(i)}/P_{MaxPM(i)} \quad (5.1)$$

The proposed clustering algorithm to group PMs is given in figure 5.2. Based on the weight value calculated using the above equation, PM with the highest weight value in each cluster is known termed as cluster head. Cluster head controls that particular cluster. The weight value $W_{PM(i)}$ decides a PM belongs to which cluster. Then it checks for the similar clusters to lodge PM and allocate it into that cluster. This clustering activity repeats whenever a new PM is active in the datacenter. This initial clustering reduces resource discovery overhead, since at the time of entry of a PM itself. It also helps to reduce searching and response time during the user request.

Algorithm: Clustering Algorithm

```
Input: PM with memory, storage and processing power
Output: Different clusters of PMs
1. Start
2. Insert a PM to the datacenter
3. For each PM
4. Calculate $W_{PM(i)}=M_{PM(i)}/M_{MaxPM (i)}+S_{PM (i)}/S_{MaxPM(i)}+P_{speed PM(i)}/P_{MaxPM(i)}$ for
   entered PM
5. Move the PM to $C_k$ if $W_{PM(i)} \geq I_{min}$ and $W_{PM(i)} < I_{max}$
6. End for
7. Return Clusters
8. Stop
```

Fig. 5.2 PM clustering algorithm

5.3.2 Energy Aware VM Migration

The next step is the energy-aware VM migration. Here each cluster is divided again into two sub clusters based on the PM weight value. The algorithm for VM migration is shown in figure 5.3. The detailed description of the algorithm is given below.

Algorithm: Energy Aware VM Allocation

```
1. Start
2. Input: VM, PMList.
3. Output: Allocation of VM to PMList.
4. Input a VM with $M_{VM(i)}$, $S_{VM(i)}$ and $P_{VM(i)}$
5. For each VM
6.    Calculate $W_{VM(i)} = (M_{VM(i)}/M_{Max VM(i)})+(S_{VM(i)}/S_{MaxVM(i)})+(P_{VM(i)}/P_{MaxVM(i)})$
7.    Select appropriate $C_k$ using *clustering( )* algorithm
8.    Consider all PMs in $C_k$
```

```
9.  Sort PMs in $C_k$ in descending order of its W
10. For each PM
11.     If $W_{PM(i)} < (I_{min}(C_k) + I_{max}(C_k))/2$
12.     Assign $PM_i$ to $C_1$List
13.     Else
14.         Assign $PM_i$ to $C_2$ List
15.         Keep idle all the PMs in $C_2$ List
16.         Best_FitVM() Allocation
17.         Consider $C_1$List
18.         For Each PM in $C_2$ List
19.             If $((M_{PM(i)} > M_{VM(i)})$ and $(S_{PM(i)} > S_{VM(i)})$ and $(P_{P(i)} > P_{VM(i)}))$
20.             Allocate $VM_i$ to $PM_i$
21.             $M_{PM(i)} = M_{PM(i)} - M_{VM(i)}$
22.             $S_{PM(i)} = S_{PM(i)} - S_{VM(i)}$
23.             $P_{PM(i)} = P_{PM(i)} - P_{VM(i)}$
24.           End For
25.     If $((M_{PM(i)} < M_{VM(i)})$ or $(S_{PM(i)} < S_{VM(i)})$ or $(P_{P(i)} < P_{VM(i)}))$
26.     Consider $C_2$ List and move all PMs from idle mode to active mode.
27.     For each PM in $C_2$ List
28.         If $((M_{PM(i)} > M_{VM(i)})$ and $(S_{PM(i)} > S_{VM(i)})$ and $(P_{P(i)} > P_{VM(i)}))$
29.         Assign $VM_i$ to $PM_i$
30.         $M_{PM(i)} = M_{PM(i)} - M_{VM(i)}$
31.         $S_{PM(i)} = S_{PM(i)} - S_{VM(i)}$
32.         $P_{PM(i)} = P_{PM(i)} - P_{VM(i)}$
33.         Else
34.             Add $VM_i$ to UnAssignedVMList
35.          End If
36.      End For
37.      End If
38. End For
```

Fig. 5.3 Energy Aware VM allocation

The average weight value is considered for cluster partitioning. So we have two clusters. The first cluster C_1 contains PMs with a weight value less than the average weight value. The cluster C_2 contains PMs with a weight value, which is greater than or equal to the average weight value. Initially, the physical machines in cluster C_2 is kept idle. When VM allocation step starts i.e. when a user request reaches the method first consider the cluster C_1 for VM allocation, and if a suitable resource found it allocate VM to it. If the user requested requirements are not satisfied by the PMs in the cluster C_1, then only the method considers PMs in C_2. If a PM in C_2 is considered and allocated then the algorithm change the status of the idle PMs to active mode. While allocating VMs to in these active PMs we have used Best-Fit allocation strategy. The formation of sub clusters during the VM allocation process will further result in energy reduction.

We have adopted the best-fit VM strategy, for VM allocation. In this allocation method, based on the processing speed we sort VMs in ascending order. For this, an available PMs list is maintained in the decreasing order of utilization. The next step is to find a PM having enough resources from the list of sorted PMs in order to allocate user request VMs from the VMs list. This allocation procedure is repeated until every VM_i in the list are mapped to host PM_j. The VMs are assigned to PMs up to maximum capacity, without degrading the processing time. Thus this algorithm reduces the number of active PMs required for assigning VMs. Since the active PMs are less, the power consumption in the datacenter will be reduced.

5.3.3 Process allocation

The users have some requirement to execute their tasks in the cloud. These requirement specifications may like memory, storage and processing power. So in order to allocate a user task to a particular VM, these requirements have to be considered. To do so, we calculate the weight value of a user-submitted process using the equation (5.2). This weight value is used to determine the cluster into which a process has to be considered.

$$W_{P(i)} = M_{P(i)}/M_{Max\ P(i)} + S_{P(i)}/S_{MaxP(i)} + P_{P(i)}/P_{MaxP(i)} \qquad (5.2)$$

From the obtained optimal cluster, then the algorithm considers the VMs in the ascending order of its weight. Then using the best-fit strategy the algorithm allocates process to the VMs in this selected cluster. The status of VMs is set to ‘1’ if it is allocated with a user process. This allocation is based on the customer requirement for speed, memory and storage parameters. When an allocation is done at the same time, the current capacity of that VM and PM are recalculated. If a process is unable to assign to a VM that satisfies customer requirements, then a list is maintained for unassigned processes to reconsider when suitable VMs are available. The detailed process allocation steps are given in figure 5.4.

Algorithm: Process Allocation

```
1. Start
2. Input: process, VMList
3. Output: Process allocation to VM in VMList
4. For each process $P_i$
5.    Calculate $W_{P(i)} = M_{P(i)} / M_{Max\ P(i)} + S_{P(i)} / S_{MaxP(i)} + P_{P(i)}/P_{MaxP(i)}$
6.    Select appropriate cluster using clustering algorithm()
7.    Best_Fit Process()
8.    Sort VMs in a cluster to VMList based on $W$ in ascending order
9.    Set UnAssignedProcessList = NULL
10.   Set processStatus = 0
11.    For each VM in VMList do
12.      If ($P_{VM(i)} > P_{P(i)}$) and ($M_{VM(i)}) > M_{P(i)}$) and ($S_{VM(i)} > S_{P(i)}$) then
13.      Assign $P_i$ to $VM_i$
14.      Set processStatus = 1
15.      $P_{VM(i)} = P_{VM(i)} - P_{P(i)}$
16.      $M_{VM(i)} = M_{VM(i)} - M_{P(i)}$
17.      $S_{VM(i)} = S_{VM(i)} - S_{P(i)})$
18.      Else
19.         Add $P_i$ to UnAssignedProcessList[ ]
20.      End if
21.      End for
22. End for
23. Stop
```

Fig. 5.4 Process allocation algorithm

5. 4 Ant Colony based Method

Then we developed an ant colony based algorithm for comparative analysis with our proposed energy-aware clustering method. Ant colony algorithm is based on probability function and its result depends on the value of the pheromone deposited by the ants during its travel in a path. Here we considered two types of pheromone values, foraging and trailing pheromone value.

In ant colony algorithms the Trailing Pheromone (TP) is defined as the pheromone which leads an ant to return it to the nest. Here when an ant is defined as an agent to find a suitable PM for allocating VM. Ant will follow the path with the maximum amount of Trailing Pheromone so that other VM request can follow this path or this PM. Foraging Pheromone (FP) is the pheromone which ant deposits when a suitable VM is found. When an ant is not carrying any VM request, it will try to find a PM with a minimum amount of FP. This is to find a PM that left from others, so ant needs to follow the minimum amount of FP unlike using Trailing Pheromone. In this method, the search space contains all the PMs available in the datacenter.

The ants continuously move in the forward direction in the datacenter network encountering the overloaded node or under loaded node. The foraging pheromone is updated using the equation (5.3).

$$FP_{(t+1)} = (1 - \theta)FP_{(t)} + \Delta(FP) \qquad (5.3)$$

Here θ is the pheromone evaporation rate. $FP_{(t)}$ and $FP_{(t+1)}$ are the foraging pheromone at time *t* and *t+1* respectively.

The trailing pheromone is updated using the equation (5.4).

$$TP_{(t+1)} = (1 - \theta)TP_{(t)} + \Delta(TP) \qquad (5.4)$$

$TP_{(t)}$ and $TP_{(t+1)}$ are the trailing pheromone at time *t* and *t+1* respectively.

5.5 Experimental Setup and Performance Evaluation

We have experimented the proposed method using 1000 physical machines in the simulated cloud environment. Each of this PM can accommodate a number of VMs with a maximum memory size of 32 GB with processing power up to 3000 MIPS. The storage capacities of each can vary up to 1 TB.

We have compared the performance of the proposed method with existing Min-Max, and Ant colony load balancing algorithms. For the performance evaluation, the parameters considered are number of PMs Searched, response time, resource discovery time, execution time, number of PMs used for VM allocation, energy consumption, total energy cost.

5.5.1 Number of PMs searched

Time needed for resource discovery is an important parameter to measure the performance of a cloud provider. Resource discovery also affects response time.

Let a cloud ecosystem with *K* number of PM which is arranged in *N* clusters. i.e., each cluster contains an average of *K/N* number of PMs. So for a resource discovery, the system has to perform minim *N* searches in the beginning. In this scenario, each cluster contains an average of *K/N* PMs. Hence, for finding the optimal PM from this cluster, the algorithm requires maximum *K/N* searches. Therefore, the total number of searches required for optimal resource discovery can be calculated using the equation (5.5).

$$\text{Total\_Number\_of\_PMs\_Searched (S)} = N + (K/N) \qquad (5.5)$$

We also compared the proposed method with Min-Max algorithm. This algorithm searches all the PMs for calculating the completion time for a job.

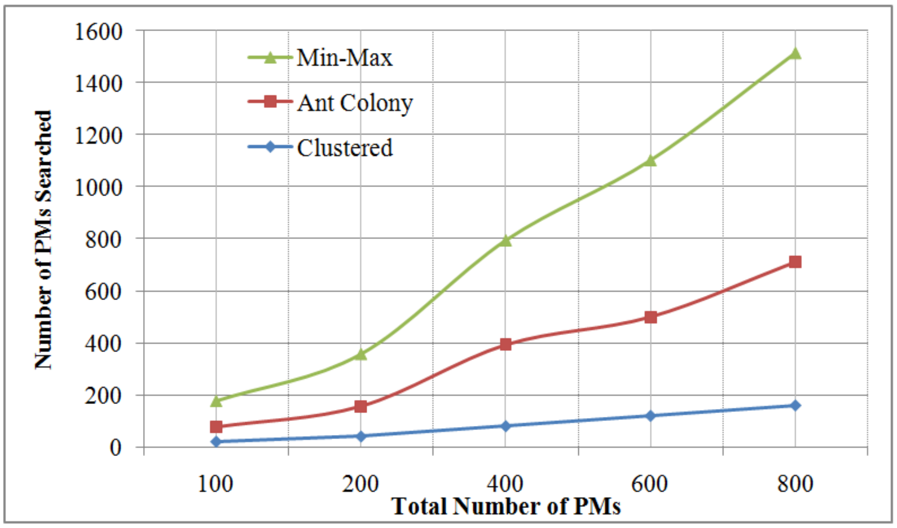


Fig. 5.5 Number of PMs searched

The experimental results for the total number of PMs searched for VM allocation are shown in figure 5.5. From the graph, we can observe that the proposed cluster oriented algorithm searches only in the most favourable cluster to find optimal VMs. It also indicates that the proposed method searches a lesser number of PMs to find VMs suitable for the customer's requirement. Accordingly this reduces the resource discovery time to find PMs for placing a particular VM with user specification.

5.5.2 Response Time

In cloud computing response time is the sum of resource discovery time and execution time. This can be represented using the equation (5.6). Resource discovery time depends on bandwidth and network traffic.

Response_Time = Resource discovery time + Execution_Time (5.6)

Here execution time of a PM can be calculated using the equation.

$$\text{Execution\_Time}_{\text{PM}} = \frac{\text{Load}_{\text{PM}}}{\text{Capacity}_{\text{PM}}}$$

Here both load and capacity is in Million Instructions Per Second (MIPS).

Table 5.2: Response time (Number of Processes = 100)

Total Number of PMs	**Clustered (Sec)**	**Ant Colony (Sec)**	**Min-Max (Sec)**
100	1.18	1.96	2.78
200	1.12	2.70	4.32
400	1.74	6.44	8.14
600	2.49	7.73	12.09
800	3.27	11.13	16.07

In the dynamic cloud environment bandwidth and speed of the internet are some of the important factors that affect overall performance. Due to fluctuating bandwidth, we have fixed 20 ms time to find a suitable PM for the experimental conditions. We have applied different loads to different number of PMs and repeated the experiment several times. Here the response time with different numbers of PMs for constant number of processes and different number of processes with a constant number of PMs are measured.

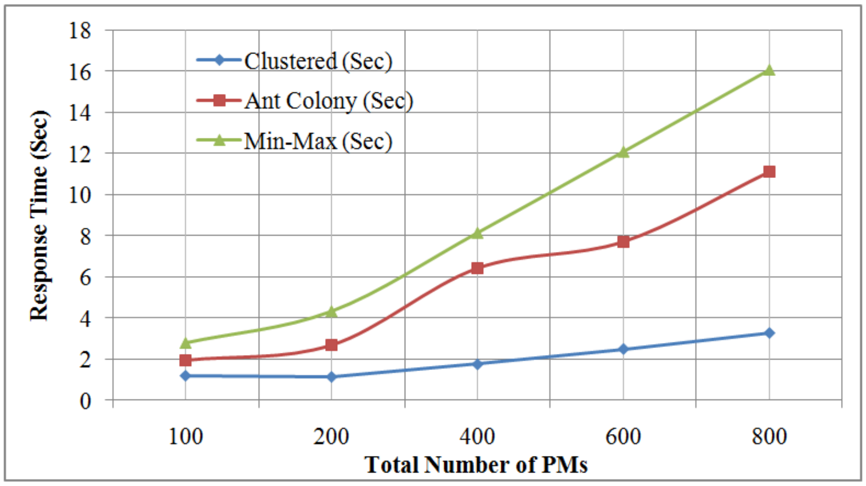


Fig. 5.6 Response Time Comparison (Number of Processes = 100)

We have fixed number of user processes as 100 and the results are tabulated in table 5.2 and the respective graph is shown in figure 5.6. This indicates that the proposed algorithm gives better improvement in response time than Ant colony algorithm and traditional Min-Max algorithm. This enables cloud service providers to provide better quality of service to their customers.

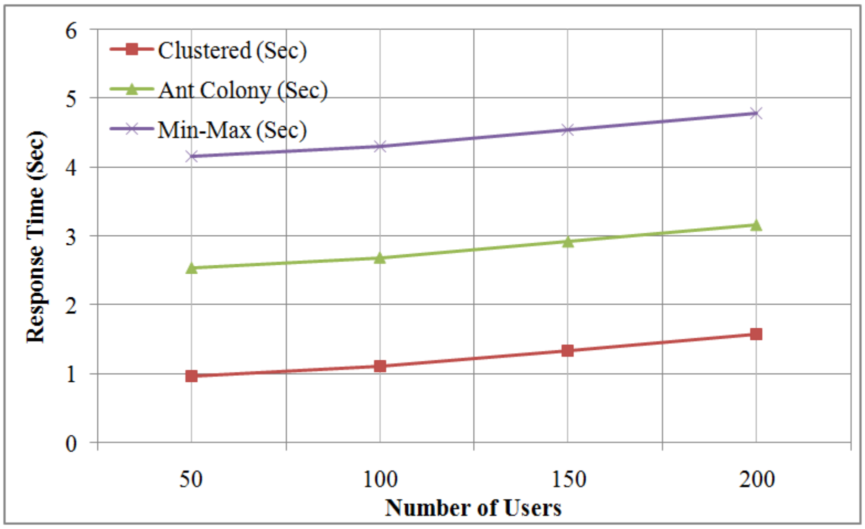


Fig. 5.7 Response Time Comparison (Number of PMs = 200)

Next, we have measured response time when the number of PMs kept constant as 200. Then we varied the number of user process and response is measured for repeated experiments. The experimental result is shown in figure 5.7. Our experimental result again shows that the proposed method improves response time compared to state-of-art methods like Ant colony and Min-Max algorithms.

5.5.3 Number of PMs Used for VM Allocation

We have conducted the experiments to know the number of PMs used for VM allocation. The results are compared with both Min-Max and Ant- Colony algorithm. From the table we can observe that the proposed system uses a very less number of active servers. This is due to the advantage of the clustering method which keeps PMs with minimum weight value is safe or hibernate mode and only activates these PMs when required. For example, when the total available PMs are 400, the proposed method uses only 30 PMs at the same time Ant colony and Min-Max use 114 and 150 PMs respectively. Similarly, when the cloud environment consists of 800 PMs, the proposed method used only 50 servers while Ant colony and Min-Max used 276 and 250 servers respectively. All these results are tabulated in table 5.3 and the respective graph is shown in figure 5.8.

Table 5.3: Number of PMs Used

Total Number of PMs	**Clustered**	**Ant Colony**	**Min-Max**
100	8	29	49
200	18	59	99
400	30	114	150
600	40	190	201
800	50	276	250

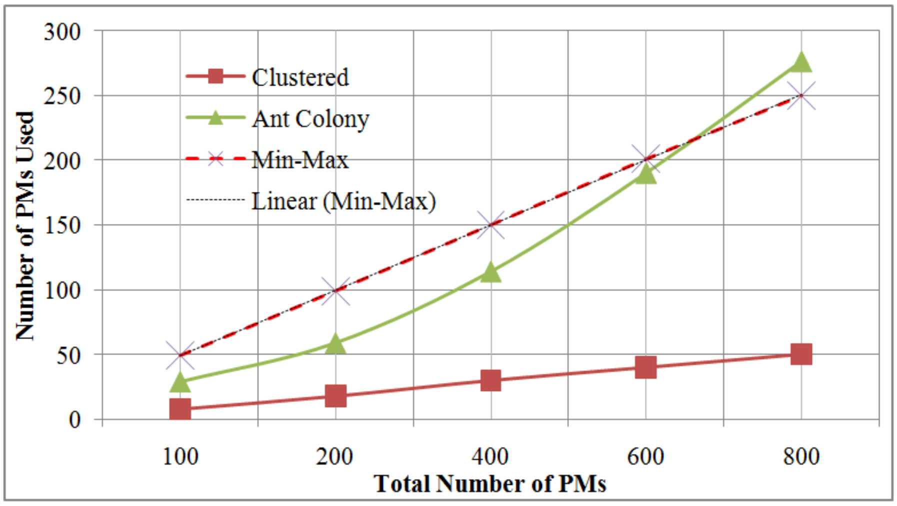


Fig. 5.8 Number of PMs used

5.5.4 Energy Consumption

We have also measured the energy consumption of servers in the datacenter. The energy usage pattern is shown in figure 5.9. The figure indicates that the proposed method is energy efficient and supports green computing. It consumes less energy since it uses only less number of active PMs for hosting VMs.

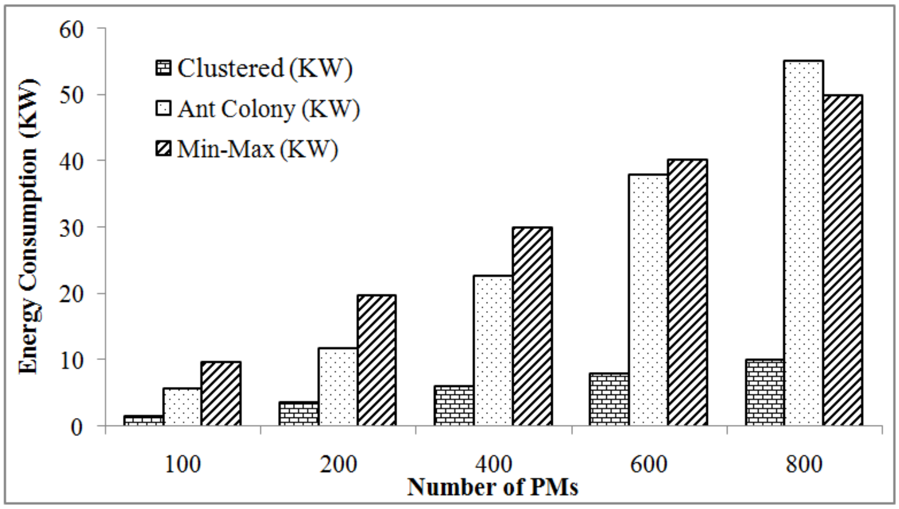


Fig. 5.9 Energy Consumption

5.5.5 Total Energy Cost

The proposed method uses, less number of PMs and this will be reflected in the energy cost. The energy price per KWh is taken as Rs.6/- unit. The cost of energy consumption is calculated based on the equation (5.7) and the results are tabulated in table 5.4.

Total_Energy_Cost/Day =(24*(Power-utilized* Energy_price)) (5.7)

Table 5.4: Total Energy Cost

Number of VMs	**Clustered**	**Ant Colony**	**Min-Max**
100	230.40	835.20	1411.20
200	518.40	1699.20	2851.20
400	864.00	3283.20	4320.00
600	1152.00	5472.60	5788.80
800	1440.00	7948.80	7200.00

5.6 Summary

We have proposed and implemented an energy-aware clustered load balancing mechanism to reduce the searching overhead for resource discovery and to improve the response time. Our algorithm also minimizes power consumption and energy cost. It efficiently uses available active servers. Thus it improves the overall quality of service in the cloud datacenters.

CHAPTER 6

ENHANCED STABILITY THROUGH INTERFERENCE AWARE PREDICTION

Contents

6.1 Introduction

The most important aspect of a computing service is user satisfaction and it doesn’t depend on whether the service is deployed in a cloud or in a non-cloud environment. In cloud frequent migrations affects system stability and thereby decrease in the quality of service delivered. Proper migration control plan is important in large scale data centers. Optimal tuning of the resource management method will avoid frequent VM migrations. This is essential in maintaining the overall system performance and quality of services delivered to the end users. A good allocation strategy should consider these factors and mitigate frequent migrations to improve the QoS offered to the customers.

Most common type of resource management techniques are based on the parameters like time (makespan, response time, waiting time, etc), energy efficiency and cost, with little attention on the interferences caused due to VM migrations. Such interferences degrade the overall performance of the system, and consequently violate the conditions in the service level agreement (SLA) between the cloud service provider and the customer. So in this chapter we focus on an interference aware prediction mechanism for VM migration, with auto scaling. A brief introduction to the automatic scaling policy is given in the section 6.1.2.

6.1.1 Interference

This work proposes an interference prediction technique for VM migration that will help in the respective auto scaling of resources. In a datacenter there are several CPU cores running simultaneously. But common resources such as memory, buses, etc., are shared among these cores. Therefore, the actual processing power cannot be achieved or used. VM data transfer due to migrations also uses these common resources, which cause overhead. i.e., migrations cause undesirable delay in computation.

Since several VMs with different applications are running in a PM, there will be performance degradation in the performance of the system due to sharing of common resources. It is also a fact that this data transfer in buses due to frequent VM migrations results in instability in the cloud eco system. All the above factors cause performance degradation or delay in computation is called interference. By reducing interference, we can achieve system stability.

In this method, interference is taken as a measure to quantify stability. Lower interference means higher stability. Also, low interference will reduce the chance for future VM migrations, thereby increasing the stability of the cloud eco system. To improve system stability VMs are migrated to physical machines having low interference.

The proposed work is intended for the stability in the performance and scalability of resources, when the user workload increases beyond a certain threshold value. So, VMs in a particular host can be migrated to appropriate destinations based on least interference values, for the performance improvement of entire cloud system. This will reduce the number of migrations in the cloud system.

6.1.2 Stability and auto scaling

Auto scaling is one of the hot features of cloud computing that facilitates resource scalability beyond datacenter boundaries. Scalability increases the performance of cloud eco system in terms of storage, processing power, throughput, and reliability. Resource scalability improves the throughput of the system without rejecting and reducing the input workload. In the dynamic cloud environment, auto scaling of resources enable cloud service providers (CSP) to satisfy customers for their computation needs, without affecting performance. When a particular PM or a CSP itself can’t cope with the user requirements, the resources are automatically scaled out to another PM or any CSP. This scaling of resources must be done as fast as possible, since any delay during the execution creates degradation in the performance. The auto scaling should be done based on some already defined threshold values [172]. When workload increases beyond the value of this defined threshold, scaling up has to occur so as to reduce SLA violations. The system must also have to release unused resources by either scale-down or scale-in process, when the workload decreases. Due to this automatic scaling process, the system avoids unnecessary VM migrations. Thus the cloud eco system can achieve stability and there by performance improvement.

6.1.3 Need for prediction mechanism

If the system can predict the interferences due to overloaded conditions, the suitable scaling decisions can be taken in advance to ensure performance. This ensures seamless execution of the scheduled user tasks. The proposed prediction model will calculate the interferences more precisely so that it will be easy to scale VM and hence maintain guaranteed SLA. The objective of this work is to make a seamless task execution during the VM migration, using a prediction model in the dynamic cloud environment with ensured SLA, and least prediction error.

6.2 Related Works

Only few papers discusses about parameters like VM bandwidth allocation and related issues in the performance. VM bandwidth allocation is one of the severe issues in maintaining good service quality. The method called Falloc [173] guarantee bandwidth for VMs based on their base bandwidth requirements and the share residual bandwidth in proportion to weights of VMs. They have simulated the system and proved that it ensure the fairness in allocating bandwidth on congested links to VMs in datacenters.

The iAware [174] is a novel live VM placement method based on demand-supply model, which try to reduce interferences. It calculates the interference in PMs based on an empirical formula and the resource demand of VM. The experiments are validated through simulated environment using realistic benchmark workloads on Xen cluster. The algorithm also tries to improve power consumption with load balancing. The final VM placement decision is based on a simple ranking method.

The paper [175] points out that multiple task execution creates interferences in the system. This article presents a 4-dimensional multiple resource model, along with a brief description about commonly happening interferences during multiple tasks. VMFlocks is an incrementally scalable high performance VM migration service designed for cross datacenters [176]. It efficiently uses the available cloud resources to accelerate data de-duplication and to transfer processes with a minimum access control.

The server consolidation and VM scale-in process create a significant variation in the power efficiency and thermal performance of distributed systems [177]. The contention of resources impacts the distributed system throughput differently, and significant variation is observed in the performance [178]. Power and cost are related factors in distributed computing. In pSciMapper [179], to maintain the required throughput rate, power, and cost analysis is performed in different iterations. Based on the experiments using real and synthetic scientific workflows, they have obtained an optimum power and cost factor.

Migration of VM from one physical machine to another allows load balancing and fault tolerance in heterogeneous cloud computing environment. The optimization method in [180] demonstrates how the migration of a running computer with its state information, can be transferred to another location. There are some common issues like delay, cost, and robustness are in live VM migration. Migrations can degrade the performance of other collocated VMs in the cloud. Paper [181] proposes a model for live migration between the source and destination to address this system noise due to migrations. The performance analysis is done based on multiple resources migrations and related migration time.

VM migrations cause a side effect called migration noise [182]. This is a kind of delay that occurs during the VM migration process, due to certain factors. An algorithm called sonic migration comparatively examines the performance of all active VMs, and reduces the migration noise created due to VM migrations.

In cloud environment, the migration of VM does not transfer host side cache state [183]. This leads to the degradation in performance of newly migrated VM, until the cache is rebuilt. To minimize VM-perceived performance degradation period before the completion of migration, a host-side cache warm-up mechanism called Successor, is used to parallelize cache warm-up and VM migrations. Cost and power prediction during the live migration of VMs is also a critical factor.

Multiple resource allocation to heterogeneous jobs having different priority, is an interesting problem in cloud computing. The work in [184] considers dynamic and non-stationary cloud environment with two classes of jobs, namely emergency and elective. In this multi-resource allocation problem, the jobs in emergency class should be performed immediately, while elective jobs have to wait for its turn. The simulation results are promising and need to be tested in real conditions. The objective of live migration is to ensure continuity in operations, with guaranteed QoS as per the agreed SLA between the service provider and the customer. This will aid system maintenance, reconfiguration, load balancing, and fault tolerance.

A virtual machine Dynamic Forecast Migration (VM-DFM) algorithm deals with the dynamic changes in virtual machine memory resource consumption [185]. An intelligent cloud load balancing technique is flexible to integrate multiple load balancers [186]. This technique can separate the allocating process and migrating process while preserving a guaranteed level of service. The QoS ensured in this method is based on the parameters such as performance, reliability security and time.

Energy aware VM scheduling based on CPU and I/O bound characteristics [187] is a move towards green IT. The simulated results in homogeneous environment showed that the reduction in the number of migrations reduces the energy utilization and SLA violations. Paper [188] presents a new security incorporated energy consumption method for task scheduling. This DVFS based method supports auto scaling for load balancing. Forecasting the resource requirement will improve the SLA necessities. Proactive scheduling approach based on fuzzy logic [189] executed on Google trace data set, analyses its effectiveness with univariate and multivariate variable approach. A practical approach towards application scaling is briefly described in [190]. The feasibility and performance of this method is demonstrated with biomedical workflow.

A Task and Resource Allocation CONtrol (TRACON) is a novel method based on application characteristics at the runtime [191]. It reduces the interference due to concurrent data intensive applications in large scale data centers. In contrast, there are several resource allocation technologies based on factors like, load, cost, power, priority and interference for maintaining QoS.

6.3 System Design

We have considered following architecture as shown in the figure 6.1 for the implementation of proposed live migration. The VM migrations are carried out by considering interferences caused due to VM migrations and the agreed SLA.

- The role of performance tracing tool is to monitor the VMs and PMs in the datacenter.

- The migration protocol comprise of two modules. First module is for VM utilization and next is a method to separate under provisioned and over provisioned VMs. The duty of this module is to identify the right VM candidate for migration.

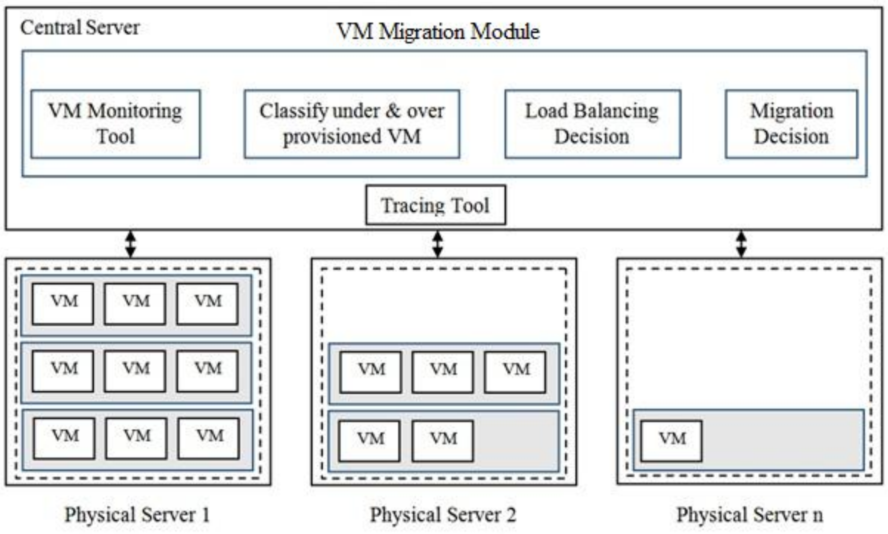


Fig. 6.1 VM live migration architecture

The aim of elastic computing is to scale the computing facility according to the workload. So it should consider and handle load balancing and the resulting interference.

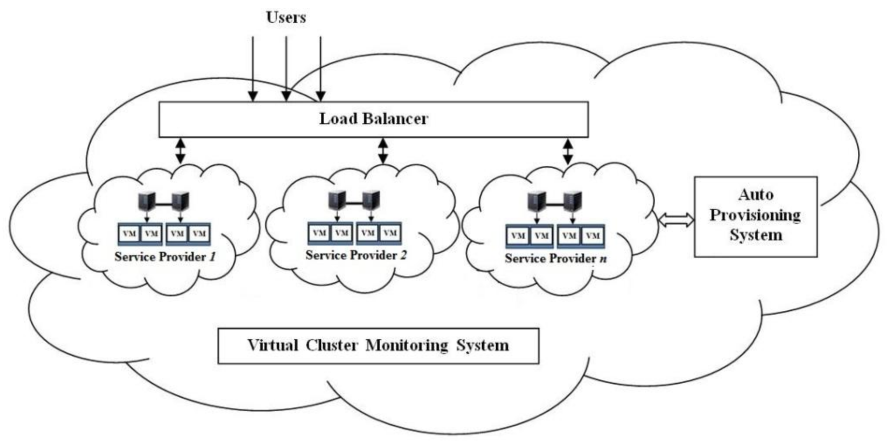


Fig. 6.2 VM live migration scalable architecture

The design of the proposed live migration architecture in intra cloud with auto scaling is shown in the figure 6.2.

Our proposed live VM migration design contains three main components namely:

1. Load balancer
2. Virtual cluster monitor system
3. Auto-provisioning system with a scaling algorithm.

Load balancer: The function of load balancer is to balance the requests between different virtual machines in a virtual cluster of a cloud service provider that perform the similar type of applications.

Virtual cluster monitor system: For effective monitoring the system must collect resource usage information of each virtual cluster. Virtual cluster monitor is responsible for the collection of resource usage information from these clusters.

Auto provisioning system: According to the workload in a cluster, it facilitates horizontal expansion or shrinking the number of VMs on that particular cluster. For example, if all the computing resources of a cloud service provider's are completely under utilization, then the auto provisioning mechanism will assign the resources to other suitable service provider transparently without any user intervention. This is done through live migration of currently running live VM to another service provider, thus the proposed method assures interrupt free service to the customers.

The proposed method calculates the VM Utilization (VMU) status using the equation (6.1).

$$\text{VMU} = \text{Total Resource Allotted} - \text{Utilized Resources} \qquad (6.1)$$

The resource utilization is periodically checked and based on this value the proposed method predicts the future resource requirements. Here we defined a server as a hot spot, if the server resource utilization is above a hot threshold value. i.e., hotspot status indicates that the server is overloaded, and hence, some VMs running on it should be migrated to other locations for ensured QoS.

Every physical machine contains several virtual machines, so we can define Physical Machine Utilization (PMU) as the sum of total number of VMUs of all VMs in a particular PM, at a particular time. The calculation of VMU and PMU is the responsibility of the tracing tool. According to these values the PMs are marked as under or over provisioned.

6.3.1 Dynamic scaling

The Amazon provides facility to the users for defining a scaling policy when their requirement increases. [192]. Regular checking of resource usage statistics is necessary for taking effective scaling decisions. This is impractical since frequent monitoring is expensive and may leads to frequent VM migrations. So an auto scaling group with policies is defined that will scale the resources with the assistance of a prediction mechanism. The mechanism fires when there is a forecast for the need of resources in future. Thus the policy automatically decides when to scale-out or scale-in and where to scale the resources. This resource scale-out or shrinkage may depend on the resource consumption metric. This resource consumption metric may be based on network traffic, CPU usage, etc.

The algorithm for auto scaling process is given in figure 6.3.

```
The auto scaling process
1. Define Metrics for VMs based on SLA
2. Monitor the specified metrics for all VM instances in the auto scaling
   group
3. Update metrics depends on the workload
4. If the metrics violate the threshold, fire alarm
5. If the system need more resources
            a.  Send Scale-in-policy message
        5.1 Otherwise
            b.  Send Scale-out-policy message
6. Receive Auto scaling policy message, and perform auto scaling for the
   auto scaling group.
7. Continue the process until user deletes scaling policies or the auto
   scaling group.
```

Fig. 6.3 The auto scaling process

6.3.2 Application interference

The physical machines in a datacenter contain many CPU cores. Theoretically we can say that the performance of a physical server increases linearly with the number of CPU cores increases. i.e., is, the overall performance of a server can be calculated by multiple of the performance one core. In reality, the performance doesn't meet the theoretical expectation. This is due to sharing of computation power for other computing related activities. E.g. consider a physical machine with 6 cores, one memory bus controller. This memory bus controller is shared among these cores. Due to the speed difference of memory and bus controller, there will be memory latency. So, the actual load on the system can be mathematically modeled as in the equation (6.2).

If a PM contains *k* cores and has *m* tasks to process

$$\mathrm{Load}_{\mathrm{Total}} = \sum_{k=1}^{n} \mathrm{Load}_{\mathrm{k}} + \gamma . \prod_{\mathrm{i,j}=1}^{m} \mathrm{Load}_{\mathrm{i/j}} \qquad (6.2)$$

Here, the second term is the parasitic load, due to interference. γ is the regression coefficient and its value is controlled between [0, 1].

E.g. Consider a situation with two loads A and B. Then the parasitic load can be represented as

$$\text{Parasitic Load}_{\mathrm{A|B}} = \gamma._{\mathrm{A|B}}.\mathrm{Load}_{\mathrm{A}}.\mathrm{Load}_{\mathrm{B}}$$

6.4 Pareto Derived Interference Prediction Model

In a datacenter, data intensive applications cause interference and this influence the performance of the services rendered to the customers. The performance of the system is mainly depends on the execution time and throughput, but VM interference affect these performance parameters. Since multiple VMs are running simultaneously in a physical machine, the actual workload is also dependent on these independent VMs. In this kind of condition, multivariate regression analysis is a good choice to analyze the problem. Hence, the problem can be modeled as a generalized multivariate linear regression model as given below.

Let Y represents the total load, here it is dependent variable and X_1, X_2, X_3, ..., X_k are the individual VM loads, here these are considered as the independent variables. Also b_1, b_2, b_3, ..., b_k are the constants and k is the number of independent variables. Then Y can be represented by the equation (6.3)

$$Y = b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k \qquad (6.3)$$

Then, the coefficient of this regression model can be obtained using the following equation (6.4)

$$b = [X^TX]^{-1}[X^TY] \qquad (6.4)$$

Where X^T stands for transpose of the matrix X

Now we have to reduce the error in the prediction. The error in the prediction can be reduced when we could minimize the Sum of Squared Error (SSE). When the SSE is minimum, then the prediction is considered as best one.

Let us consider that there are only two active VMs are in a PM, and then the model can be represented using the equation (6.5).

$$Y = \sum_{i=1}^{k} \mathrm{b}_1 \,.\mathrm{X}_{\mathrm{vm}\,1,i} + \sum_{i=1}^{n} \mathrm{b}_2 .\mathrm{X}_{\mathrm{vm}\,2,i} \qquad (6.5)$$

Where b_1 and b_2 are constants that normalize the prediction error.

The accuracy of the prediction model increases when there is increase in the number of VMs. The objective of this model is to choose the most favorable threshold range for the user to carry out the operation, without compromising the agreement between the provider and the customer. Using this proposed prediction mechanism provides an optimum threshold range for the operation with the guaranteed SLA.

6.4.1 Pareto optimality

Single objective optimization problems are quite easy to solve, usually it have only one optimal solution. A multi-objective optimization (MOP) problem contains many conflicting objectives that need simultaneous optimization of these objectives. Since these objectives are conflicts each other, usually there will not be a single optimal solution. Therefore, for the decision making, is cumbersome task and may require a tradeoff between different solutions from the finite set of possible solutions by making negotiations. In this tradeoff, the improvement of one objective comes at the expense of another objective. In this kind of multi-objective optimization situations Pareto optimality [97, 144, 145] is a good choice.

Pareto principle: The aim of Pareto principle is to converge the solutions to the Pareto front and then find the diversified solutions scattered over it.

"According to Pareto principle a set of non-dominated solutions, is optimal, if no objective can be enhanced without sacrificing at least one other objective. i.e., a solution α is termed as dominated by

another solution β if, and only if, β is equally good or better than α with respect to all other objectives".

In multi objective or multi attribute optimization problems, the Pareto set or Pareto front is a subset of the set of feasible points or solutions. This set contains all the points or solutions with atleast one objective optimized, while holding all other objectives as constant. For the conflicting objectives, there may exist some near optimal solutions in the Pareto front segment. In Pareto-front region, these conflicting objectives will behave in a non-conflicting manner and optimal solutions are from this region. So this region is also known as Pareto-optimal region. We can say that the set of solutions converges to a Pareto front in this optimal region. Near optimal solutions can easily be identified from the Pareto-optimal front, since the number of objectives are less than the actual dimension in this region. This justifies that Pareto method is a good candidate for multi objective optimization problems.

6.4.2 Pareto-derived interference aware (PiA) algorithm.

Generally Pareto method is a two step procedure. At first, it converges to the Pareto front and next it discovers a solution set from the possible points of solutions sprinkled over the Pareto front. The figure 6.4 shows the proposed Pareto-derived interference aware (PiA) algorithm.

In this algorithm Y is the total load value for all the tasks within each PM. $\gamma_{i,j}$ is the load factor for each task within VM$_j$ using equation (6.1). Here linear interference prediction model is based on equation (6.2).

The weighted sum or scalarization method allows to combine multiple objectives into a single objective scalar function. We have used this concept to model the multi-objective cloud task scheduling to a single scalar function.

Algorithm: Pareto-derived interference aware (PiA)

```
Input Data: Targeted VM$_j$, where $j \in 1, . . . , n$; and
Resource pool consisting of PM$_k$, where $k \in 1, . . . , m$;
Collect load factor $\gamma_{i,j}$ for each task within VM$_j$ using equation (6.1);
//Here model is linear interference prediction model in equation (6.2)
Output: Schedule VM$_j$ to PM$_k$ assignments
1. For $j = 1$ to $n$ do
2.     for $k = 1$ to $m$ do
3.         $Y_j$= Predict($\gamma_{i, j}$, PM$_k$)
4.     end for
5. end for
6. Apply Pareto ranking
7. Select {Dominant Pareto set from $Y_j$ }
8. PM$_{candidate}$ = Min$_j$ ($Y_j$)
9. Assign (VM$_j$, PM$_{candidate}$)              //Assigns VMs to candidate PMs
```

Fig. 6.4 Pareto-derived interference aware algorithm

Let us consider a scheduling problem with n number of virtual machines and our main aim is to minimize the interferences caused during VM migrations. Then, the weighted-sum method that minimizes a positively weighted convex sum of the objectives can be represented as

$$min = \sum_{i=1}^{n} \alpha_i . f_i \, (x) \qquad (6.6)$$

where $\sum_{i=1}^{n} \alpha_i = 1$ and $\alpha_i > 0,1, \dots n$ and $x \in X$

In the equation (6.6), X is the set of all VMs in the platform, and i represent the weight vector. The term $f_i(x)$ is the objective function in the total interference. Then the total interference can be represented as the sum of migration (M_i) and co-location interferences (N_i). Hence the function $f_i(x)$ can be simplified as in equation (6.7),

$$f_i(x) = \sum_{i=1}^{n} (M_i + N_i)\ ;\ \forall i \qquad (6.7)$$

6.5 Experimental Setup and Analysis

We have tested our above method using web service and a parallel processing application in the cloud computing environment. The web service is chosen because it should available at any time and should provide the fastest response time, regardless of the number of users served. So we can test the dynamicity of the proposed scaling mechanism. Since cloud allows super computer level computing facility by distributing the work parallel into several nodes. Usually the users unaware about the exact number f computing nodes they are utilized to complete their jobs. When VMs processing user requests by observing QoS parameters, there is equal role for the interference awareness in such a scenario. The proposed method VMs that cause lesser interference are selected for assigning user tasks.

6.5.1 Experimental conditions

We have tested the proposed prediction mechanism in Gungoos cloud environment with following specifications as shown in table 6.1. We have chosen Gungoos [204] as our test platform since it is a powerful cloud provider with strong sever features.

Table 6.1: Experimental Conditions

CPU Specifications	Dual 15 Core Xeon Haswell (total 30 cores) with 2 x 24 MB cache processors
HDD	8 x 2 TB SSD hard drives mounted on RAID for the database 4-12 TB disk drive arrays and total 128 GB of RAM
Environment	Hadoop, SQL and XAMPP

6.5.2 Analysis

In order to prove the effectiveness of the proposed method, we have done comparative analysis with the traditional First Fit Decreasing (FFD) [193] and iAware [174] prediction algorithms. To develop experimental setup and environment, we have used Hadoop, SQL and XAMPP. During the live migration total VMU is measured to analyze the migration statistics.

The experiment is designed in such a way that the proposed method predicts the interferences at different threshold ranges. Here we have adopted threshold ranges 55-60%, 60-65%, 65-70% and 70-75%. The objectives are defined in terms of VMs or the threshold range at which the operations are carried out. Here Pareto optimality is defined as changes to different VM task allocations that makes at least one individual task execution better off, without making any other individual VMs worse off. As indicated earlier an allocation becomes Pareto optimal when no further Pareto improvements can be made to that allocation.

6.5.2.1 Threshold range

The interference were measured for different threshold range. The experiments were conducted with different number of VMs in each threshold range as specified above and corresponding interference were recorded. The value for the range 55-60% threshold is shown in the table 6.2. The figure 6.5 shows the respective Pareto graph for the above threshold. From the table and figure we can observe that when the number of VMs increase, the value of the interference also decreases. The interference value reaches 3 for the threshold range 55-60% when the active numbers of VMs are 9 and 10. This shows we can't further improvement in the interference value beyond this point and hence the method converged to the Pareto optimal solution at this range.

Table 6.2: Pareto table for threshold range 55-60 %

Virtual Machines	Interference	Cumulative %
1	36	19.93 %
2	35.5	39.59 %
3	29	55.65 %
4	23	68.38 %
5	18	78.35 %
6	15.1	86.71 %
7	10	92.25 %
8	8	96.68 %
9	3	98.34 %
10	3	100 %

Similarly, the experiments were conducted for threshold ranges 60-65%, 65-70% and 70-75% and the respective interferences are shown in the figures 6.6, 6.7 and 6.8 respectively.

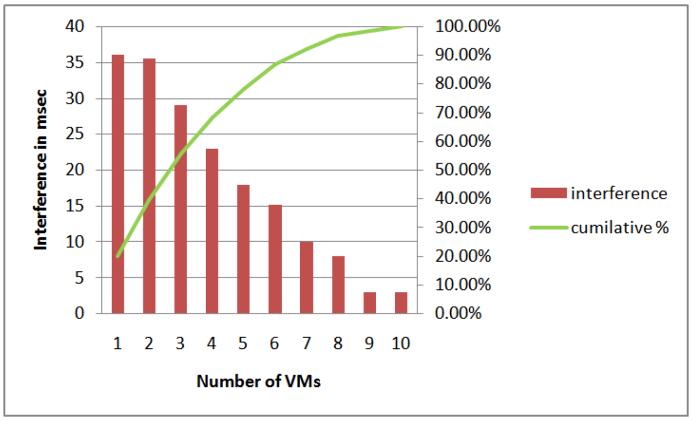


Fig. 6.5 Pareto graph for threshold 55-60 %

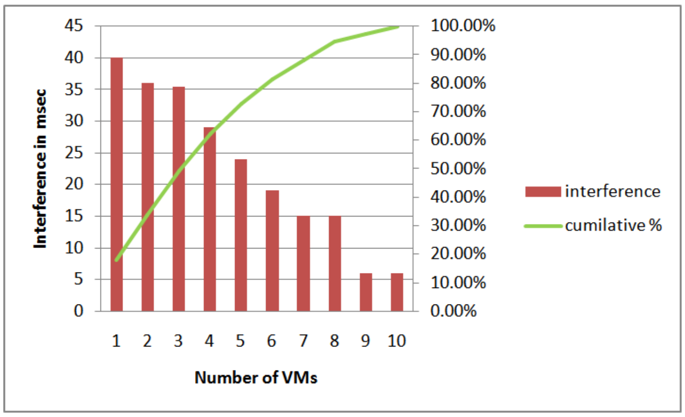


Fig. 6.6 Pareto graph for threshold 60-65%

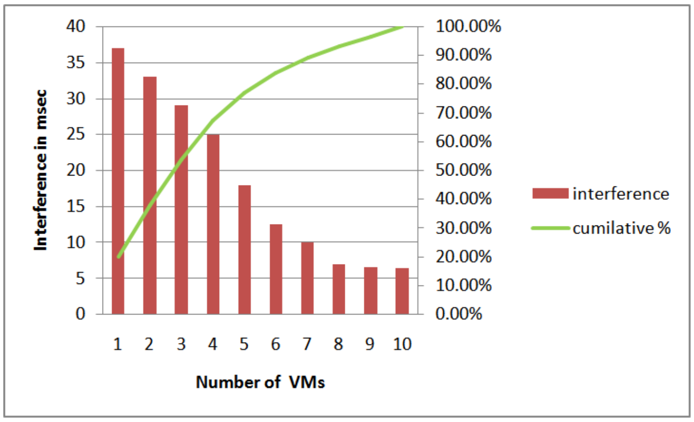


Fig. 6.7 Pareto graph for threshold 65-70 %

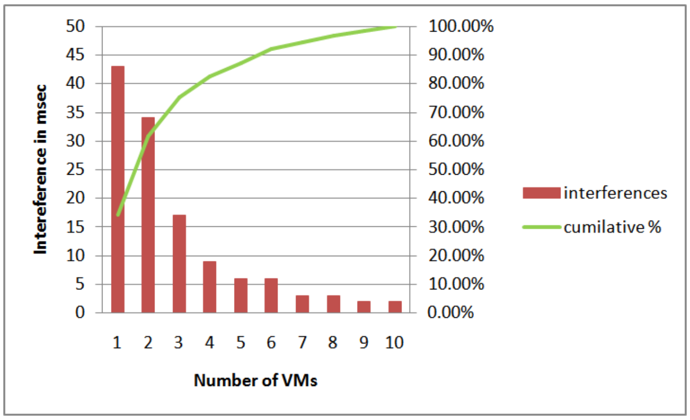


Fig. 6.8 Pareto graph for threshold 70-75 %

According to the above results, we can say that the proposed interference aware prediction model predicts the most precise threshold range with SLA violation free or with very less violation operation. So this method mitigates the interference caused due to VM migrations.

6.5.2.2 Prediction error

Standard error is the measure of accuracy of predictions done by the method. Let I_a is the actual and I_P is the predicted interferences, then it can be calculated using the equation 6.8.

$$\text{Prediction Error} = \sqrt{\frac{(I_a - I_p)^2}{n}} \qquad (6.8)$$

Where *n* is the total pair of observations.

The comparison of prediction errors at different threshold range are shown in the table 6.3. From the table we can observe that threshold range 65-70% gives less prediction error. At this range the error in the prediction is only 3.706. Hence we showed that the proposed PiA method predicted the accurate threshold range with minimum interference. So the provider can chose this threshold level for SLA violation free operations. The figure 6.9 shows the graphical representation of prediction errors in different threshold ranges with different number of VMs.

Table 6.3: Comparison of prediction errors at different threshold range

Virtual Machines	**70-75 %**	**65-70 %**	**60-65 %**	**55-60 %**
1	43	37	40	36
2	34	33	36	35.5
3	17	29	35.5	29
4	9	25	29	23
5	6	18	24	18
6	6	12.5	19	15.1
7	3	10	15	10
8	3	7	15	8
9	2	6.5	6	3
10	2	6.4	6	3
SD	14.5697	11.7195	12.3433	12.4660
Standard Error	4.6073	3.7060	3.90331	3.9421

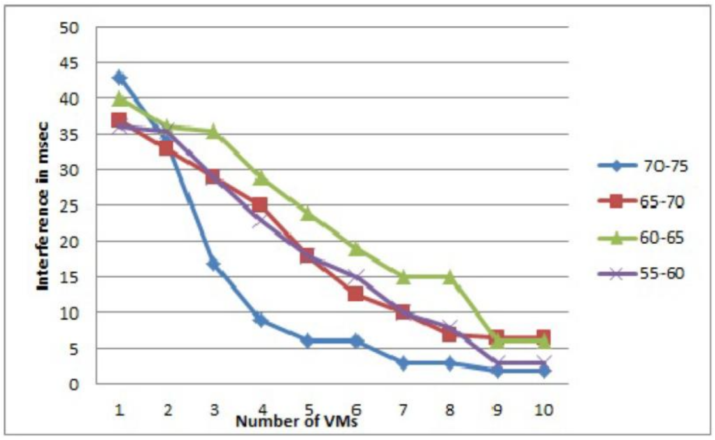


Fig. 6.9 Comparison of prediction error among different threshold ranges

6.5.2.3 Comparative analysis of interference

Again comparative study was conducted to evaluate the delay caused in the performance with other existing algorithms.

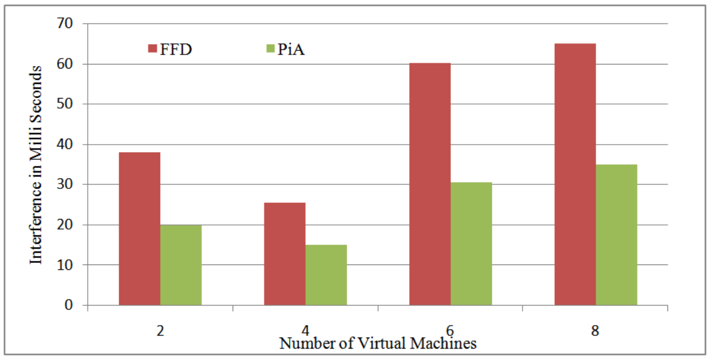


Fig. 6.10 Comparison of interference with First Fit Decreasing (FFD)

We have compared the delay with FFD algorithm with different number of VMs and the graphical representation is shown in the figure 6.10. The results showed that there is significant improvement in the interference, due to the mechanism adopted in the proposed PiA method.

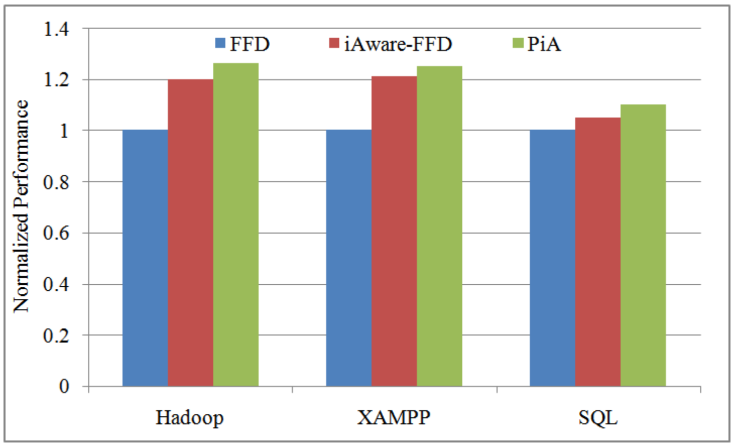


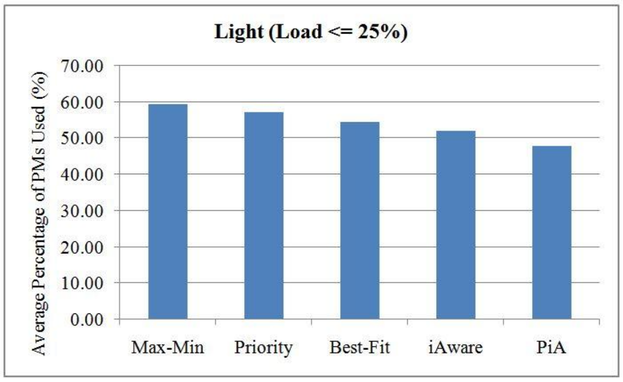
Fig. 6.11 Performance comparison

Again in order to prove the efficiency of the system, we have compared the proposed method based on the workload. For this, we have used two VMs with Hadoop, XAMPP and SQL. The performance comparison is done with traditional FFD and iAware FFD and it is given in the figure 6.11. The iAware-FFD and proposed PiA method have nearly equal performance improvement compared to the FFD method. In the Hadoop environment PiA gives 5% normalized performance improvement than iAware and with XAMPP the corresponding improvement is 3.31%. We can notice that in the case of SQL, the performance improved by 4.76% than iAware and 10% than the traditional FFD methods. Overall in all the test cases, our proposed PiA method outperforms than FFD and iAware-FFD. The low performance of traditional FFD is that since it works based on greedy approach for VM consolidation. Due to the greedy nature, the physical servers have to accommodate more number of VMs, which causes severe interferences among VMs. While the PiA method, consider only less interference PMs for VM placement, hence the better performance.

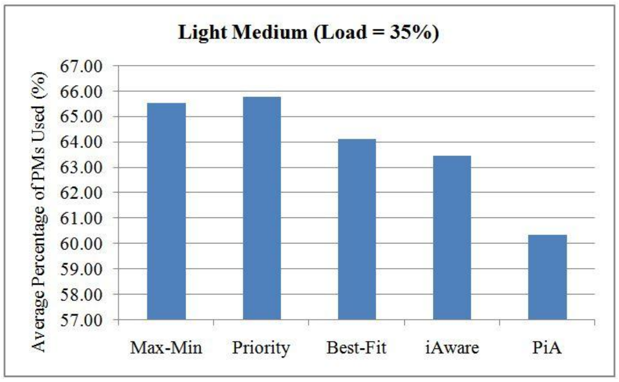
6.5.2.4 Number of physical machines used

The utilization of the resource can be measured based on the number of physical machines used for placing the VMs. We have tested the system in different load conditions. (a) light load, where 25% of load is applied to the system, (b) light medium load, where 35% of the load is applied as input user load, (c) with 50%, considered as medium load, (d) heavy load, where the input is able to utilize about 75% of the processing capacity of the cloud and (e) an over booked stage, where any input which is greater than 75% of the processing capacity of the entire cloud is utilized.

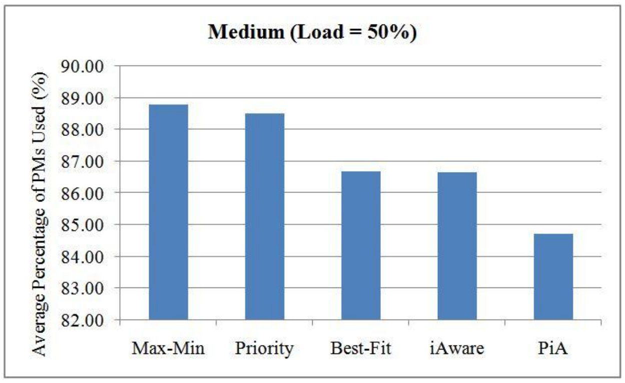
The comparative study in the five different load conditions is shown in the figure 6.12. The experimental results showed in the figure indicate that the PiA method uses less number of physical machines to place the requested VMs in all the scenarios with minimum interference.



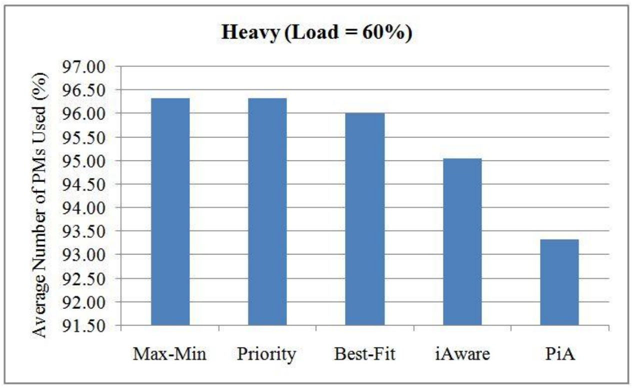
6.12 (a) Light load



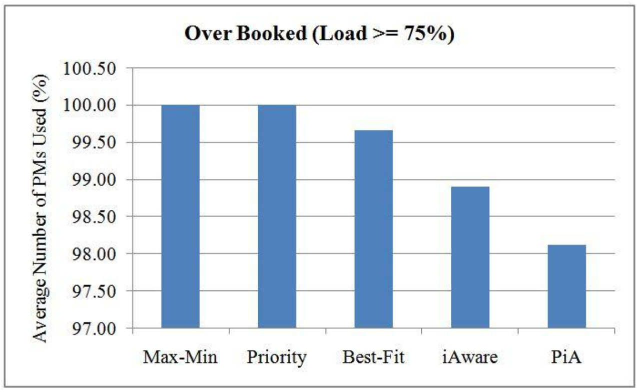
6.12 (b) Light Medium load



6.12 (c) Medium load



6.12 (d) Heavy load



6.12 (e) Over booked stage

Fig. 6.12 Average Number of Physical Machines used in different conditions

While analyzing the figures following improvements are happened. In light load, it used 12% lesser number of PMs than the traditional Max-Min algorithm [194] and with priority [195] method is 10%. While with Best-Fit the improvement is 7% and with iAware it is nearly 4%. We can observe similar result improvement in light medium and medium loads.

In heavy load conditions our method used 3% less number of PMs than Max-Min, Priority and Best-fit algorithms. While comparing with iAware, the proposed method used 1.73% less number of PMs.

When we increased the load to very high leading to an overbooked stage, Max-Min and Priority methods used all the available PMs in the environment. Even in this severe load condition, our method kept 0.78% PMs in idle condition. This indicated that the PiA method effectively and efficiently used all the active PMs.

6.6 Summary

In this chapter we have proposed an interference aware prediction mechanism for resource management in the cloud. The proposed live migration architecture comprises of load balancer, virtual cluster monitor system and an auto provisioning system with a scaling algorithm. The proposed PiA method forecasts the interference value accurately and predicted the optimum threshold range for efficient operation so that service provider can manage the SLA requirement requested by the customers. In the case of increased demand, with the help of prediction values, the auto scaling policy scale the resources to meets the user requirements with assured quality. So the proposed method helps in the seamless computing by predicting the accurate threshold range. The performance of the method is tested in real time cloud environment and the prediction accuracy is verified by calculating the standard error in predictions, in different threshold ranges. Again the comparative analysis was done with other methods such as FFD and iAware-FFD in Hadoop, XAMPP and SQL environments. This test results also prove the effectiveness of the proposed PiA method. The major reason behind this improvement is that the proposed method always searches for less interference PMs for VM placement thus it reduces VM migrations and achieves stability. Again the method is tested in five different workload conditions to know the resource utilization. The experiments results confirm that the PiA method efficiently utilizes the active PMs than other state-of-art algorithms.

CHAPTER 7

SLA ENFORCEMENT WITH AUTO SCALING

Contents

7.1 Introduction

The cloud service vendors offer a vivid variety of purchasing options and dynamic prices to the customers. They announce spot instance prices in the market-oriented cloud to attract more customers, increase the resource usage and earn more revenue. To incorporate these purchasing/promotional offers/dynamic prices, a good scheduling method is needed. Also, methods are needed to ensure whether these offers are maintained by the service provider through an SLA enforcement mechanism.

The violations in SLA will degrade the performance of the service provider and thereby decrease the credibility of them among customers. So to cope with the service level conditions, sometimes the providers have to increase the resource capacity by scaling out within the same datacenter or in a co-located datacenter for a satisfying marketing option with their consumers. An efficient scheduling algorithm should consider the demand from the clients for resource provisioning and de-provisioning. Since the resource demand and price varies with time, there needs efficient scheduling mechanisms for optimal allocation of resources to the user workload.

Cloud computing provides purchase provision to enhance 24x7x365 support and monitoring, trust, security understanding of business and customizable service at a lower cost. The service providers like Salesforce.com [8], Amazon [10], Rackspace [13], etc., promote their customizable services among the business enterprises by offering promotional offers. With the intention to increase their income and attract more number of customers, these promotional offers are based on their current and historical resource utilization rate and cost-benefit analysis. The hybrid technology supports customers to acquire host and on-premise offers along with other cloud offers to promote on-demand infrastructure and to reduce the operational cost. Dynamic provisioning with elastic computing facility and SLA enforcement is still a problem to be addressed.

In this chapter, a Petri Net model is proposed and used for scheduling the tasks based on user requirements and to incorporate dynamic spot prices. Here SLA is ensured with the help of auto scaling mechanism. In this model, the SLA requirements considered are CPU speed, memory, makespan and bandwidth with a fewer number of virtual machine migrations.

Our experimental results indicate that the proposed system efficiently performs dynamic provisioning and elasticity in multiple public clouds with scaling that reduces makespan, number of SLA violations, penalty cost and maximizes profit with the help of auto scaling mechanism.

7.1.1 Petri Net

Petri Net is a promising mathematical modeling tool for describing distributed and parallel systems. It is a good tool for the representation of distributed and parallel information processing systems that are characterized by concurrency, asynchronous, non-deterministic or stochastic [196].

In cloud, scheduling user requests to the available VMs with minimum completion time is considered as an NP-hard problem. In this type of situation, Petri Net models are one of the good methods to obtain optimal results [32, 196, 206]. Here we have proposed a Petri Net based for resource allocation with auto-scaling.

7.1.2 Spot Instances

Amazon provides a type of prices instances called Spot Instance (SI) to sell the idle time of its EC2 data centers [10]. It is a rebated pricing model used by service providers like Amazon to sell their spare resource capacity using an auction method in the open cloud market. This price is based on the demand-supply pattern at real-time.

This spot price history is freely provided by AWS per SI [197] for each data center and also available through other third parties such as Cloudxchange [239]. For the experimental purpose, we have taken the spot instance price from AWS. Figure 7.1 shows the historical average normalized price of Amazon EC2 for the instance type c1.xlarge for a day.

Even though spot instances allow opportunity to use unused server capacity of a service provider at a lower price, there is a need of efficient algorithms for SLA enforcement for the interrupt free service.

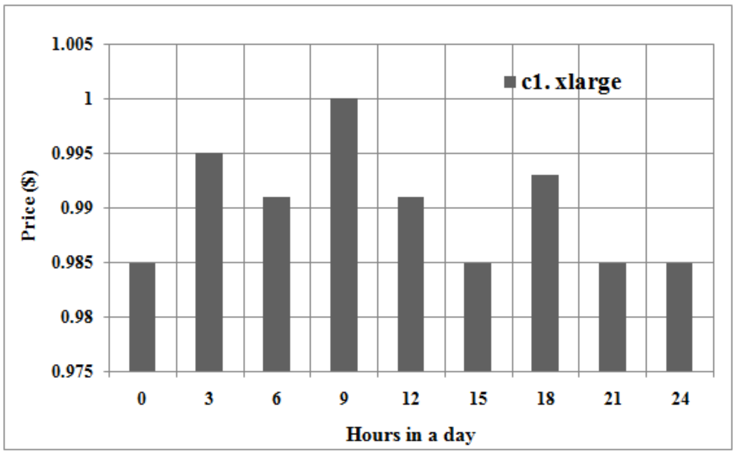


Fig. 7.1 Normalized average spot instance price of c1.xlarge for a day

7.2 Related Works

From the chapter 2 it is understood that the researchers proposed several scheduling techniques for allocation of tasks, which are focused on different parameters such as makespan, load balancing, power consumption, delay, cost and profit. In this chapter, we consider some cloud management policies used for scheduling and related issues in the market oriented cloud. Even though these techniques have used different scheduling policies but some issues are to be addressed deeply and needed to be fine tuned. The specialty of the market oriented cloud is that a customer has an opportunity to bid the price to hire a service. Usually, service providers' offer price is based on the historical bidding details and spot instance prices. The CSPs like AWS defined a spot price as a bidding strategy to maximizing their revenue.

Auto scaling helps rapid provisioning and de-provisioning of resources with minimal management effort or service provider interaction [3]. Open Cloud Computing Interface (OCCI) standard provides an effective resource management in Service – based Business Processes (SBPs). Elasticity can adapt the oscillating workload in the cloud and ensuring QoS by using an autonomic loop called MAPE (Monitor, Analyze, Plan, Execute). This is an autonomic infrastructure that supports optimized resource utilization and save the cost [198]. Scheduling of resources on multiple clouds will enhance the performance with minimum operational cost and time. In paper [20] the workflows considered as a sequence of transactions with multiple micro tasks provides minimum completion time but no SLA enforcement mechanism.

A lot of auction based mechanisms are proposed for business users. Mechanisms like continuous double auction [199, 211], knowledge based double auction [212], combinatorial [36, 213] and negotiation model [214] are the some of them comes under this category. These methods didn’t consider spot instance and historical data about price calculation for resource allocation. It is also noted that these methods don't support auto scaling.

In the auction based dynamic scheduling [199] VM resources are indexed based on the requesting time, loading time, dealing time. It also considers the cost of the service based on minimum affordable price policy considering both client and service provider.

Heuristic methods such as PSO [200] that consider communication and data transmission cost for workflow scheduling. Time and load balancing issues are not considered in this method. All the above methods discussed so far suffer from frequent migrations, which increases system imbalance in the cloud ecosystem and thereby degradation in the overall performance. Modified version of Dynamic Voltage and Frequency Scaling (DVFS) [215, 216] have efficient energy consumption by lowering the frequency, but it increases the makespan that leads to SLA violations.

The paper [206] presented highlights and tools used for workflow scheduling using Petri Net theory. Petri Net has format semantics and has many analysis techniques. It can be used for both event and state based systems. Since cloud scheduling is state based, Petri Net is good for cloud task scheduling.

Sever consolidation in the cloud [201, 202] reduces the number of active physical machines so that it harnesses green computing. Sometimes improper server consolidation mechanisms result in frequent migrations and it causes system imbalance in the cloud. So these methods have to consider QoS constraints mentioned in the SLA and the migration overhead. Unfortunately, most of these methods neglect the effect of imbalance caused due to frequent migrations. So we have proposed a method to respect QoS constraints and reduces imbalance using Petri Net.

The PreAnt policy discussed in [40] schedules the heterogeneous resources in the cloud with minimum cost as well as energy consumption. Here a fractal-based mathematical prediction model allocates service requests to VMs in an optimized way to reduce the energy as well as migration time. However, they have classified incoming requests into four different categories to map with ant colony algorithm, which causes additional time overhead while processing the requests.

The adoption of online marketing strategy provides bidding option for customers to access the service with affordable charges. The spot instances and price offered by the providers is based on the auction mechanism which enables customers to opt for online purchasing facility of services with minimum cost. Usually, the spot price is calculated based on the pricing strategies of other providers and real-time conditions [31]. The Petri Net based multi-criteria decision framework in the cloud generates a cost effective marketing option using spot instances in VM resource scheduling. The scheduling of resources in multiple clouds accelerates the service quality and low cost with maximum utilization of resources [204]. Their simulated results prove that the better cost saving can be achieved through spot instances with auto scaling mechanism but frequent migrations affect performance.

The market oriented hierarchical scheduling strategies [203, 209, 210] provides significant improvement in QoS constraint resource allocation. From the experimental results, we can see that the cost and time optimized policies will potentially increase the budget.

A scalable self scheduling scheme [207] is another method for large scale cloud systems that minimize the communication overhead. These kinds of systems are only suitable for scientific workflows.

In this section, we discussed different techniques used for scheduling tasks using the auction mechanism and issues in it. Most of these techniques are focused on the minimization of makespan. A multi-objective criterion based scheduling is needed to solve state-of-art problems in the cloud. Auto scaling with less number of migrations is also another performance indication. From these observations, a Petri Net based scheduling algorithm can support multi-criterion with contradictory requirements, which can perform auto-scaling with less number of migrations and cost saving.

7.3 Petri Net for Cloud

The aim of this work is to develop a model based on Petri Net to enforce SLA with cost-effective resource scheduling in the market oriented multi-clouds and with the minimum number of migrations. The proposed model supports multi objective decision making strategy in the allocation of service requests to the market oriented cloud.

The parameters considered are response time, makespan and cost of computation. These primary parameters are used for calculating the cost saving is cost-benefit ratio and penalty. If the bid price is higher than the spot price updated by service provider, then service is accepted with SLA negotiations for other parameters. While billing, penalty is computed based on the cost per number of SLA Violations. Then auto-scaling is carried out based on the cost-benefit-penalty calculation. The bid price is varied in real-time depending on the demand, processing power of requirement and power utilization of active servers.

7.3.1 Basics

In multi objective scheduling problems, Petri Nets are the adequate method to model complex dynamic situations. Since cloud scheduling is an NP hard problem, Petri Net modeling is a good promising solution to model dynamic cloud task scheduling model. It is a mathematical modeling language using directed bipartite graph. The task model in Petri Net can be described as follows.

A task model is represented by the tuple $PN_{Task} = (P, T, A, M_o)$, where place $P = \{p_1, p_2, \ldots.., p_n\}$ is the set of physical locations, $T = \{t_1, t_2, \ldots., t_m\}$ is the set of transitions, *A* is set of connection between location and transitions and is represented as $\{(p_i, t_j), (p_j, t_i)$ and $M_o \in \{R^+ \cup, 0\}^{|P|}$ is the initial marking.

In Petri Net models the places represent states, conditions or resources that need to be available and met before an action can be carried.

7.3.2 Principle of locality and reduced imbalance

In order to reduce frequent migrations, we have used the principle of locality. The behaviour of Petri Net can be formulated using rules for transition to occur. For transition enabling conditions and consequent actions, the Net considers only immediate vicinity of a transition.

Principle of locality states that

> *"For a successful transition depends only on local states of the locations in its immediate vicinity. Also, a successful transition changes only the local state of locations in its immediate vicinity".*

The above property of the Petri Net will reduce the frequent migrations in the cloud ecosystem, and resultant effects are reduced imbalance and better response time since the migration of tasks are based on the capacity of nearby resource specifications.

7.3.3 Petri Nets for cloud scheduling

The Petri net model for cloud scheduling is a seven tuple, $PN_i = (P, T, F, W_i, M_o, C_i, D_i)$ where, $P = (p_i |\ i = 1, 2, 3, \ldots, 11)$ is a finite set of places. The graphical representation of the proposed model for the cloud using Petri Net is given in figure 7.2. The detailed descriptions of each place is given in table 7.1,

$T = \{t_i |\ i = 1, 2, 3, \ldots 10\}$ is a finite set of transitions, where, the descriptions of transitions are also given in table 7.1,

$F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs,

$C_i = \{(dt_{ij})\} \cup \{(ET_{ij}, CT_{ij})\} \cup \{(a_i)\}$ is the set of colours, where dt_{ij} is the deadline of task j and, ET_{ij} is the expected execution time of task j at machine i, CT_{ij} is the expected communication time of task j, a_i is the current status information of machine *i*, including available VM's number and time that is ready for executing next task at that machine.

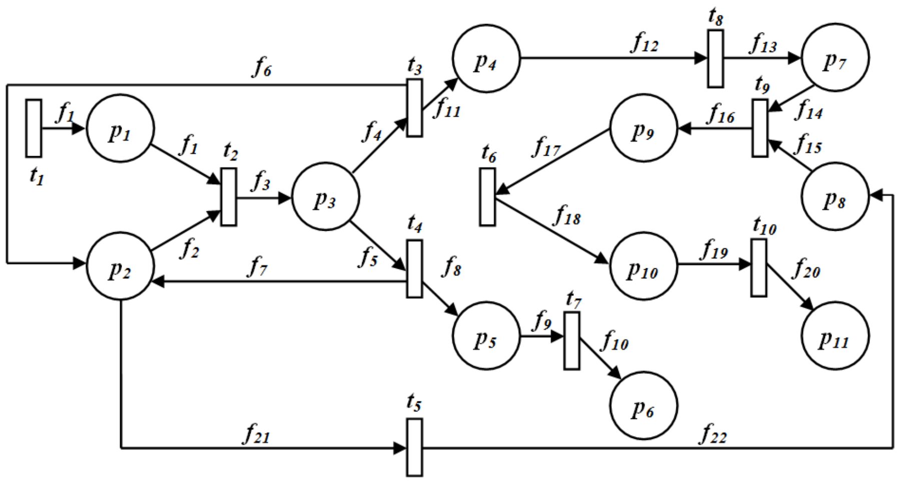


Fig. 7.2 Petri Net model for cloud scheduling

$W_i = f(F_i) = \{f_j \mid j = 1, 2, 3, \ldots\ldots 22\}$ is a finite set of weighted functions of arcs, where,

$f_1 = (dt_{ij}, ET_{ij}, CT_{ij})\}$ processing requirements of task *j* submitted at physical machine *i*.

$f_2 = (a_i)$: the information of physical machine *i* in a CSP,

$f_3 = f_4 = f_5 = f_1 + f_2$,

$f_6 = f_2$,

$f_7 = (a_i')$: the updated status information of physical machine *i* in a CSP,

$f_8 = f_{11} = f_{12} = f_{13} = f_{14} = f_1$,

f_9 = the task selected by a home scheduler,

f_{10} = completed task by a home scheduler,

f_{15} = (a_k): the current status information of resource at a remote datacenter k.

$f_{16} = f_1 + f_{15},$

f_{17} = the task selected by a home scheduler according to the algorithm,

f_{18} = the task completed by home machine,

f_{19} = Submitting completed remote/home task,

f_{20} = Sending completed task to parent datacenter,

f_{21}, f_{22} = current information of resources.

$M_o(p_2) = (a_i)$.Others places having no tokens in the initial marking.

D: $T \rightarrow R$ is a firing time delay, where, $D(t_1)$ is a random number.

$D(t_2)= D(t_3) = D(t_4) = D(t_5) =D(t_{10}) = 0.$

$D(t_8) = D(t_9)$ = communication time of task.

$D(t_6)= D(t_9)$= execution time or task or sub task.

The service request scheduling is done according to the spot price defined by AWS then the client can bid the price for servicing their request. For auto scaling the spot instance price is also a factor. This results in a considerable increase in performance and cost saving. While scheduling incoming tasks, the model checks the minimum execution time, waiting time and number of SLA violations, since the Petri Nets can check multi-criterion in the decision making. The scheduling of tasks is performed based on the number of tasks at a particular time and resource availability. Scale-up and scale-down are implemented to reduce the cost, energy and also power consumption. For auto scaling the proposed model considers a number of migrations along with penalty and profit. The system acquires better performance in terms of profit for service providers and with less number of SLA violations and minimum makespan and cost for the users. Thus the proposed model ensures the quality of the service.

Table 7.1: Description of Petri Net Places and Transitions

Place	**Description**	**Transition**	**Description**
p_1	Incoming user tasks	t_1	Submitting a task to a datacenter/CSP
$p_{2/}\ p_8$	Information about home/remote cloud resources	t_2	Collecting resource information
p_3	Ready to schedule a task	t_3	Not able to complete task with SLA requirement
p_4	Ready to schedule a task at remote datacenter/CSP	t_4	Able to complete with SLA in current datacenter
p_5	Ready to execute a task	t_5	Current status of resources
p_6	Task completed	t_6	Executing assigned remote/own tasks
p_7	Submitted tasks to remote datacenter	t_7	Executing assigned own tasks
p_9	Ready to execute remote/own tasks	t_8	Sending tasks to remote CSP's local scheduler
p_{10}	Completed remote/home tasks	t_9	Task assignment to remote local scheduler
p_{11}	Submitting remote task to home datacenter	t_{10}	Submitting completed remote/own tasks

7.3.4 Scaling process

When the demand increases to satisfy customer requirements, auto scaling with migration is required. If the demand is very low, then idle servers have to be switched off. If a scheduled request needs more processing power, using scale-up mechanism additional power can be allocated. Figure 7.3 shows auto scaling algorithm.

7.3.5 Evaluation parameters

We have evaluated the proposed model based on makespan, SLA violations and profit. The execution time of each task can be found using the equation (7.1) given below.

$$ET_{ij} = \frac{Task\ Length_i}{Processing\ \ Power\ \ of\ VM_j} \tag{7.1}$$

Where ET_{ij} is the execution time of i^{th} task on j^{th} VM.

The possible reasons for SLA violations are due to situations like deadline violations, changes in cost, processing power requirement, etc. Anyway, the violations depend on the QoS parameters specified in the agreement. The cost of computation or the profit of the provider depends on SLA violations.

The profit of the provider is the difference between service provisioning cost (C_p) and cost incurred due to enforcement of penalties (C_t) in SLA violations. This can be calculated using the equation (7.2).

$$\text{Profit}= \sum_{\lambda\in\{User\}}(C_p\ (\lambda) * C_t(\lambda) - \sum_{\omega\in\{Dl,Rt,Cost\}}(\Psi(\omega) * V_p\ ) \tag{7.2}$$

Where ω is the set of SLA parameters associated with a service λ, *Dl* represents deadline and *Rt* is the Response Time. The term $\Psi(\omega)$ is the number of SLA violations detected and V_p represents the penalty associated to the respective violations.

Algorithm: Auto scaling process

```
//Let ST represents Service Task and R represents Resource list.
For all $Task_i \in$ ST and resource $r_k \in$ R
    If ($UP_{activeservers} < PP_{servicetasks}$)   then  //UP – Utilization Power &
                                                      PP –  Processing Power
      Add Resources $r_k$  and allocate task $Task_i$ to $r_k$
      Update ST and R
    Else if ($EFT_i \geq Dl$) and ($Task_i \in ST$)  /*Dl is the Deadline & $EFT_i$ is the
                                                    Estimate Finish Time of the $i^{th}$ task */
      Scale up resources
      Update ST and R
    Else if ($UP_{activeservers} > PP_{servicetasks}$) or ( ($Task_i$  not in ST) or ($EFT_i < Dl$)
       Scale down
       ST = ST + {$Task_i$}
       Allocate task $Task_i$ to $r_k$
       Update ST and R
     Else
        Set the active server to  idle mode
     End if
End for
```

Fig. 7.3 Auto scaling process

7.4 Experimental Setup and Performance Analysis

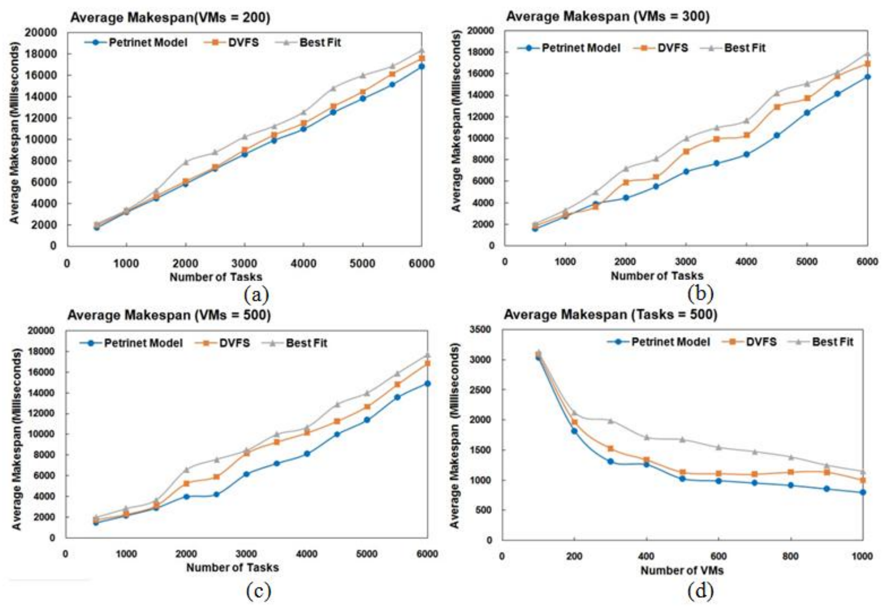
We have simulated the proposed model using CloudSim [129]. The model is tested with different parameter settings and experiments were repeated several times. The makespan, number of SLA violations, cost saving and number of migrations are measured and compared with existing methods to evaluate the effectiveness of the proposed system.

The customer gives service request with requirements such as memory, speed, requested time and bid price and the resource manager in the cloud broker analyze these requests with the current resource availability. The resource manager collector is able to access the information of each task from SLA monitor, scheduler, and cost analyzer. Accordingly the resource manager updates the current status of the tasks and resources. The proposed method is compared with Dynamic Voltage and Frequency Scaling (DVFS) and best fit algorithms.

7.4.1 Makespan

We have conducted experiments and the makespan is measured for different conditions. The makespan is collected for both an increasing number of VMs and an increasing number of tasks. As expected in the case of increasing the number of VMs, the average of makespan is reduced. The experiments were conducted with a fixed number of VMs as 200, 300 and 500 for a varying number of tasks. Figure 7.4 shows the graphical comparison of the makespan of the proposed method for a varying number of tasks and VMs with DVFS and Best-Fit algorithms. In figures 7.4 (a), (b) and (c) we can see that there is a significant reduction in makespan compared to DVFS and Best-Fit for the Petri Net method in different experimental conditions.

In figure 7.4 (d) when the number of VMs is increasing, then the rate of average makespan is decreasing. In the initial part of graph 7.4 (d) there is a small variation after it goes linear decrement. All the above results show that the performance of Petri Net model is better than state-of-art algorithm such as DVFS and Best-fit. This is due to auto scaling mechanism which causes a low execution rate because scale up and down policies in the scheduling process need some time that causes an initial delay.



The Fig. 7.4 Average makespan when VMs number is fixed (a) 200 (b) 300 (c) 500 (d) Number of tasks fixed = 500.

7.4.2 SLA Violations

Violations in the service level conditions will lead to performance degradations in QoS and thereby penalty is to be imposed. A good resource management method will always try to maintain the conditions mentioned in the SLA. One of the objectives of the proposed method is to reduce the number of violations in SLA.

We have measured the extensions that happened in the time depended parameters and difference in the cost. The auto scaling mechanism adopted helped to lower the rate of SLA violations compared to DVFS and Best-Fit algorithms. We have fixed number of VMs as 200, 300 and 500. In each of the above condition, we increased the number of tasks and SLA violations are measured. We can observe that violations are gradually increasing in all methods due to high load. The violation rate for the proposed method is very low compared to other algorithms. This proves the credibility of the proposed method. We can also observe decrement in the violations when the number of VMs are increased for a fixed number of tasks. The analysis shows that Petri Net based system produces 99.87% efficiency compared to DVFS and Best-Fit policies. This is due to auto scaling and the adoption of Petri Net's principle of locality feature. Figure 7.5 shows the graphical analysis of the number of SLA violations in different scenarios.

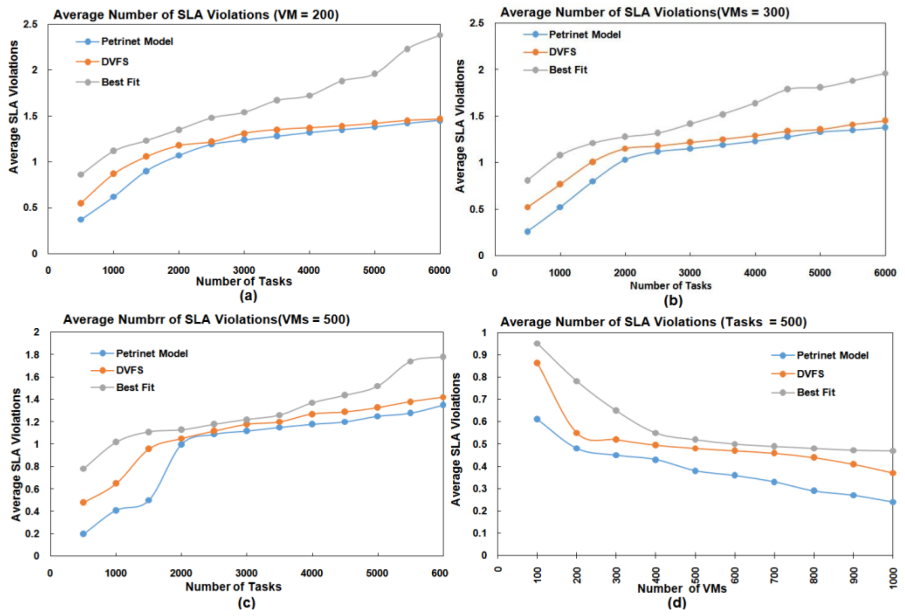


Fig. 7.5 Average number of SLA violations in different scenarios.

7.4.3 Profit

The bidding method gives customers the opportunity to choose the best services at affordable price. The customers are trying to select services at low cost and minimum makespan while the providers try to attract more customers with different attracting offers to increase their profit. The experimental results indicate that the average profit earned with Petri Net based scheduling policy is higher than DVFS and Best-Fit after consideration of penalty due to SLA violations. The penalty is also lower in the Petri Net model than the other two methods under comparison. The figure 7.6 shows the graphical representation of average profit earned when the number of VMs is fixed as 500.

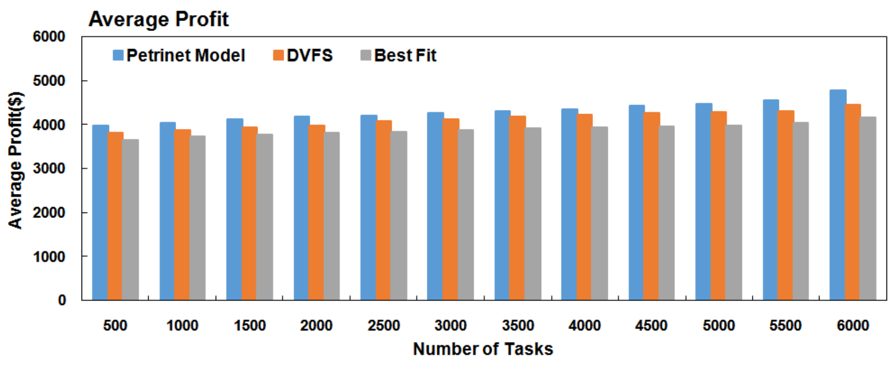


Fig. 7.6 Average profit when number of VMs is 500

7.4.4 Migrations

We have further investigated the number of migrations happened during resource management. This factor is measured to know the system stability. Even though the migration procedure is used to maintain conditions in the service agreement, but sometimes frequent migrations create an imbalance in the cloud ecosystem and hence affects overall system performance. The experiments were conducted to measure the number of migrations in low and high loads. We have assigned 500 – 3000 tasks in low load and 3500 – 6000 tasks in high load conditions. Figure 7.7 (a) show the number of migrations happened when the number of tasks are increased in low condition. From the figure, we can observe that when the load increases there is a linear increase in the number of migrations in both methods. In the proposed method the migrations is less than the existing VM selection and VM placement approach [205].

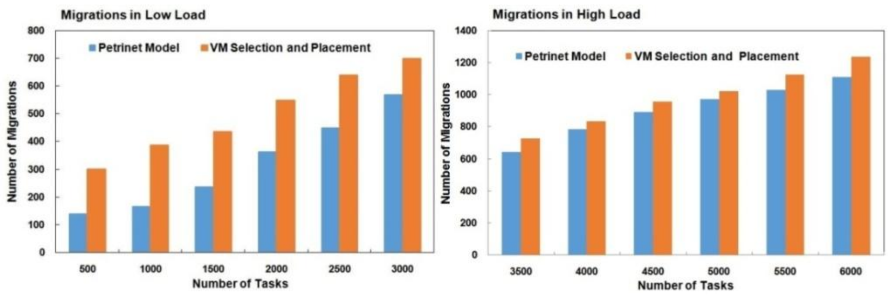


Fig. 7.7 Migrations when 200 VMs (a) Low load (b) High load

While coming to high load conditions, the number of migrations is gradually increased when the load increases. Figure 7.7 (b) shows the result of number of migrations in high load when the number of VMs is fixed as 200. To increase the resource utilization and reduce power consumption, later arriving tasks are scheduled to VMs that have already completed their assigned tasks. Proposed algorithm compared with VM selection and VM placement approach [205] and shows an average of 7 % of performance improvement in high load than later.

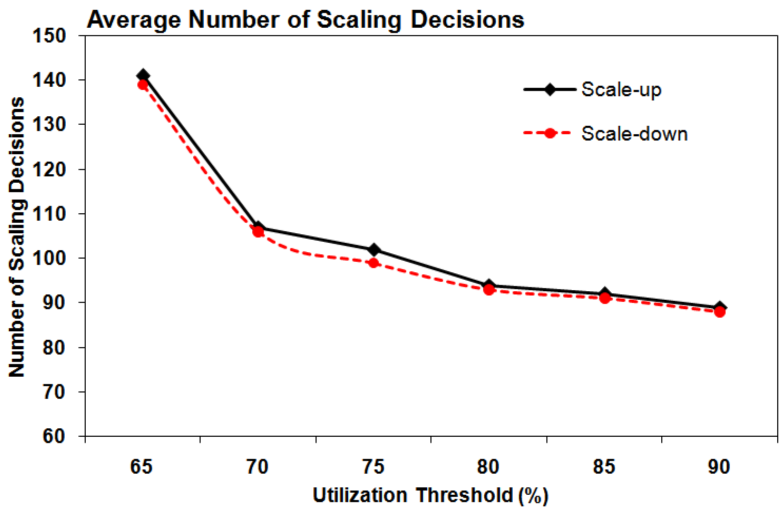


Fig. 7.8 Average number of scaling decisions

We have measured the average number of scaling decisions that happened in different CPU utilization threshold ranges. The experiments were conducted at threshold ranges 65% to 95% with a limited number of VMs. The obtained results are graphically represented in figure 7.8. If there is a chance of violation detected due to any reason, scaling-out is carried out. It also monitors the condition of all the servers and takes scale-in decision if it is found idle. The experiment results show that when the threshold is kept low, the number of scaling decisions are high and for high workload, the number of decisions is low, which shows better control over the workload and resource. This minimizes frequent migrations and hence related delay and imbalance in the cloud.

7.5 Summary

In this chapter, we have proposed scheduling and load balancing mechanism based on Petri Net model with auto scaling. Price variations, violations in deadline and response time are the major factors in SLA violations in market oriented cloud. In the proposed Petri Net model, the properties of Petri Nets are used to enhance the multi objective cloud scheduling mechanism. This model supports multi-criterion scheduling with cost saving in the dynamic market oriented cloud. Here we considered dynamic spot pricing strategy to test the proposed method in multi-clouds. Penalty is computed based on the number of violations occurred in the agreed conditions. Multi-criterion actions coordinate in the resource manager and finally, the optimal result is delivered to customers with minimized cost and makespan. This resource-task allocation is based on the historical data about the providers’ offers and current bid price for a particular service. The algorithm supports auto scaling to ensure QoS and power saving. To reduce frequency migrations, the algorithm applies locality principle, this reduces system imbalance and better load balancing. By simulation experimental results and comparative analysis, we have shown that the proposed model provides better performance in terms of time, cost and migrations.

CHAPTER 8
INTEGRATED APPROACH TOWARDS QoS

Contents

8.1 Introduction

In cloud scheduling preservation of conditions in the Service Level Agreement is essential to maintain Quality of Service (QoS). There are several scheduling methods in the cloud computing, that independently handles multi-tenant, on-demand and elastic, but integrated methods are necessary to improve the performance. Due to dynamic nature of the workload and resource availability static methods are not good for optimal scheduling. In this circumstance, a usage prediction method will help to reduce SLA violations by forecasting the future resource requirement, so that provider can arrange required resources to maintain QoS. As mentioned in previous chapters, frequent VM migrations are also a critical factor that affects quality of service delivered. Proper resource prediction will minimize VM migrations. By considering all these factors this chapter proposes an integrated SLA enforcement scheme that will consider makespan, migrations, SLA and cost with the aid of a prediction model. The incorporated prediction model is based on the past usage pattern and forecasts future SLA violations due to fluctuating workload. Based on this forecasts appropriate load balancing and scaling decisions are carried out, which reduced cost, makespan and SLA violations.

Efficient resource management is required for the effective utilization of high-end computing resources. In cloud environment VMs are operating in an isolated environment so that, it can be easily migrated to other hosts, therefore load balancing through scheduling is a good solution. Most of the scheduling mechanisms in distributed systems based on load balancing are trying to use all the hosts in the system to maintain SLA. This increases the energy as well as the operational cost.

SLA enforcement is crucial since cloud is a utility type service just like electricity or water supply. The proposed method predicts the probability of SLA violations and penalty due to it. Thus this method enforces SLA by applying penalty for SLA breaches. This method also improves system stability due to scaling mechanism by limiting frequent migrations. The experimental results show that our proposed system achieves better QoS delivery in the cloud scheduling.

8.1.1 Load Resource Allocation

The general load resource allocation architecture under consideration is given in figure 8.1, which contains the 3-layer cloud organisation. The datacenter resource manager is responsible for deploying user tasks into these physical machines. Cloud broker is the mediator between user and the provider. Most of the cloud task assignment methods are random or round robin based algorithm. This inefficient assignment results in the wastage of valuable CPU cycles of physical servers. Sometimes, heavy load will cause over utilization of some physical servers, so that the tasks assigned to those physical servers are in starvation or results in the decline of service. The under utilization of PMs affects providers revenue while over utilization results in the degradation of requested QoS and finally results in the violation of SLA. The load imbalance will degrade overall performance of the cloud.

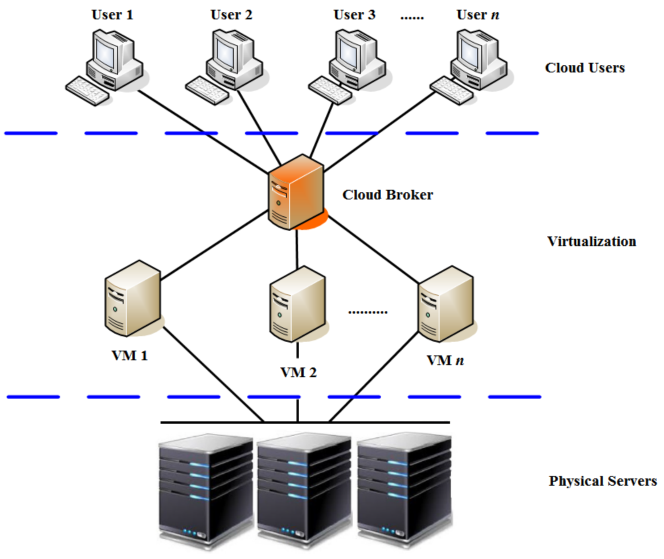


Fig. 8.1 Load resource allocation architecture

8.1.2 Role of SLA

As the applications are moved from dedicated customer premise hardware to the cloud, these applications need to achieve same or more demanding levels of services as provided by the classical installations. Therefore in cloud computing, SLA plays a key role to ensure the rights of customers. Cloud SLA is a contract between the CSP and the service consumers. In this agreement the service provided or requested is formally defined. It may contain the details about the type of service delivered, its scope, responsibilities of both parties and quality of the delivered services between the CSP and the service users. Cloud providers' resources span across multiple datacenters. SLA depends on the features of the datacenters managed by the service provider. Thus SLA is purely service based contract, and it is offered by the service providers and not a user dominated agreement. Several works of SLA negotiation have been conducted

[231, 233]. Usually SLA consider datacenter characteristics. For better performance, the network parameters are also critical factors at customer side. SLA monitoring and enforcing penalty are also crucial in maintaining QoS [222]. Here we have considered VM bandwidth, VM MIPS rate and RAM capacity as parameters for SLA.

8.1.3 Prediction Model

A good prediction mechanism will help in the proposer task-resource allocation. So this chapter proposes a prediction model based on the past usage pattern and aim to provide optimal resource management without the violations of the agreed service level conditions in cloud datacenters. It considers SLA in both initial scheduling stage and in the load balancing stage. Also, it looks into different objectives to achieve minimum makespan, minimum degree of imbalance and the minimum number of SLA violations.

The symbols used in this chapter are given in table 8.1. Rest of the chapter is organized as follows. Section 8.2 reviews different kinds of load balancing and scheduling techniques in cloud computing. Mathematical modeling and proposed method and its architecture described in section 8.3 and 8.4 respectively. Experimental results and analysis are given in the section 8.5. Finally, this chapter concludes with section 8.5.

8.2 Related Works

Load balancing and scheduling are the critical tasks in cloud resource allocation. In datacenters the user requested VMs are mapped on the physical hosts. There are several numbers of physical hosts in a datacenter; where a pool of VMs is created in these PMs based on the user requests. This VM resource pool contains VMs with different specifications.

In the dynamic cloud environment task assignment problem can be considered as a NP hard problem [224]. Finding an optimal task assignment and load balancing in the dynamic cloud environment is a cumbersome task. The optimal task deployment increases the customer satisfaction and provider's revenue. Majority of the research works concentrated are either on load balancing or scheduling. These researches are based on makespan, delay, cost, power consumption and load. Some of the methods are discussed in the following paragraphs.

A novel Weighted Signature based Load Balancing (WSLB) algorithm [218] finds that, the load assignment factor for each host in a datacenter and maps the VMs according to that specific factor. In this method, the highest configuration host has maximum load assignment factor and lowest one has less and so on. WSLB reduces the average response time in homogeneous cloud environment but load accumulation will result in SLA violations.

The geographical load balancing [217] for datacenters without prior knowledge is a good solution but it has delayed execution time due to allocation or migration at remote datacenters. Uniform load sharing is another solution proposed for load balancing. The Modified throttled algorithm [131] is based on this idea and has improved response time, compared to the existing Round-Robin and other throttled algorithms, but it considers only execution time.

The task migration technique used in [219] can improve the response time and implement parallelism of tasks in computing clusters. The limitation of this method is the high computational cost and overhead at the time of scheduling. QoS based geographical load balancing is used to overcome the impact of short-term overload on multiple clouds. It delivers acceptable QoS even in the case of resource failure and flash crowd. In this method as in [220] high monitoring overhead causes performance degradation.

Load balancing by considering the current status of all the available resources will solve the problem of inefficient utilization of resources. A scalable distributed loop self-scheduling scheme [27] is a load balancing method with reduced communication overhead. Even though the system is scalable, it is only for the homogeneous clusters.

Heuristic algorithms are sufficient for providing near optimal solutions for dynamic NP hard problems in a reasonable time. The modified intelligent water drop algorithm [221] is one among such attempt to solve workflow scheduling in computational cloud to minimize the makespan and cost. These kinds of algorithms are only capable in providing near optimal solutions.

Swarm intelligence based algorithms like Ant colony [91, 135] can be used for load balancing and scheduling [89] in cloud. It works on the basis of pheromone deposition. A node with minimum load is attracted by most of the ants. Consequently the maximum deposition of pheromone develops at that particular node and thereby, the performance is improved. Slow convergence to the optimal solutions is one of the major limitations of Ant Colony based algorithms.

In Resource Intensity Aware Load balancing (RIAL) [73] method, the VMs are migrated from overloaded Physical Machines (PM) to lightly loaded PMs. Here resource weight is determined on the basis of resource intensity. In a PM, a higher-intensive resource is assigned with a higher weight and vice versa. RIAL achieves lower-cost and faster convergence to the load balanced state, and minimizes the probability of the future load imbalance, by considering the weights when selecting VMs to migrate out and selecting destination PMs. It suffers from frequent migrations and affects the overall performance.

Cloud partitioning based load balancing model presented in [137] is simulated for public cloud, using a switch mechanism. In this conceptual framework, a switch mechanism is used to choose different strategies for different situations. This algorithm applies game theory to the load balancing strategy but it creates inconsistency in the system.

A simulation study about the SLA aware placement of VMs in elastic cloud services was done in [231]. This elastic services placement problem (ESPP) focuses on the profit maximization of service providers. The authors' tried to generalize the ESPP to a multi-unit combinatorial auction based method. This algorithm creates frequent migration that causes imbalance in the cloud.

Resource management using reinforcement learning with aggressive provisioning [225], optimality [20], and green scheduling [229] are some methods that could address the resource allocation problem. It is suitable for the rapidly increasing workloads especially in a homogeneous resource environment. Genetic Algorithm [226] based on the heuristic approach was successfully implemented for dynamic dataflow scheduling. Q-aware [230] is a QoS metric oriented workload classification and scheduling mechanism. They have minimized cost as well as time while considering QoS requirements for a class of workload. In this method, the number of migrations is high which can cause system instability.

The latency aware method [70] is able to reduce both the power and latency in cloud but has no proper workload management and load balancing mechanism. The task prioritization and financial criteria based load balancing mentioned in [52] offers a general model to adopt variable cost with improved resource utilization. SLA monitoring with corrective measures in performance as well as the cost is not incorporated in this method. Energy aware load balancing method [235] focuses only on energy conservation in homogeneous clusters. The hierarchical method based on Petri nets [236] considers only resource utilization rate and cost.

When the number of tasks is increasing the struggle for resources also increase and this creates complexity. The prime aim of scheduling algorithm is to speed up the execution of a task in cloud. The load balancer is responsible for assigning tasks intelligently to virtual machines considering the current workload and available processing power. Thus, the load balancer optimizes the resource usage, minimize execution time and avoid overloaded conditions. SLA oriented service delivery scheduler should consider server capability to meet the customer requirements, especially time requirement. To maximize the resource utilization, most of the schedulers try to allocate more tasks to a server, which will lead to overloaded conditions and subsequently SLA violations. Therefore, scheduling through load balancing is a good method to cope with SLA requirements. The methods proposed so far based on this idea consider only single parameters and lacks SLA. The limitation of two level load balancing with scheduling [237] is that it considers only makespan, while the scheduling method presented in [238] based on load balancing considers only migrations.

We have considered various other reviews [227, 228] about taxonomy on scheduling algorithms. Besides this we have conducted a detailed comprehensive review on recently proposed papers on quality of the service scheduling and load balancing techniques in the cloud. From this we can conclude that the existing mechanisms considers time and cost to deliver quality service. All the above facts point out that, there is scope for further improvement in scheduling and load balancing procedures. So here we are proposing a model which uses past usage pattern for predicting the resource requirement for optimal load balancing to reduce violations in service level agreements.

8.3 Problem Formulation

The main goal of cloud computing is the low cost computation with customer requested QoS. For this, the optimal VM allocation with load balancing is necessary. Due to the dynamic nature of the cloud environment, the scheduled jobs rarely concur with the expected execution time. Hence, it requires some sort of intelligence to assign the jobs to the optimal VMs to meet the expected QoS.

Table 8.1: Description of symbols

Symbol	**Description**
R	Minimum amount of extra resources to a VM
C_{ij}	Cost of execution
t_i	Reserved minimum MIPS for a VM at the time of creation
μ_i	Average number of MIPS requested by a user *i*.
σ_i	Standard deviation of the number of MIPS requested by user *i*
μ_{PM}	Average processing capacity of the PM
PT_{VM}	Processing time of a VM
L_{PM}	Current load on a PM
C_{PM}	Capacity of a PM
α	Cost for SLA violation
β	Cost for Service rejection
S	Total processing power of a host
$\dot{P}$	Penalty for SLA violations
σ_{PM}	Standard Deviation of load in PM

Usually user expectations are low makespan time, delay and cost of computation. The customers are expecting a service that meets the agreed service conditions or sometimes, something above it. These multiple objectives often conflict each other, so that, getting an optimal solution is a cumbersome task. It is also noted that the violations in service conditions will degrade customer satisfaction. Time and cost are two critical requirements which conflict with each other, since in terms of processing power faster resource is more expensive than slower one. Considering all these factors, the cloud task scheduling problem can be formulated as below.

- Let $i = 1, 2, 3, ..., m$ represents task indices and $j = 1, 2, 3, ..., n$ is for VM indices.

- T_{ij} denotes the time to execute the i^{th} task at j^{th} VM;
- C_{ij} stands for the cost of execution of i^{th} task in j^{th} virtual machine;
- X_{ij}= 1 if task i is assigned on machine j; 0, otherwise.
- $\dot{P}_{ij}(k)$ is the penalty associated in executing a task i on j^{th} machine for k^{th} SLA condition. Here $\dot{P} \geq 0$.
- w_t is the workload that contains number of independent tasks at time t.

So that the scheduling problem can be represented as

$$\text{Minimize} \quad \sum_{i,j=0}^{n} \left((X_{ij} T_{ij}) * C_{ij} + \dot{P}_{ij}(k) \right) \qquad (8.1)$$

Subject to the following conditions

Maximize workload w_t without any performance degradation.

Minimize cost and penalty

Minimize Degree of Imbalance (DI) in the cloud.

The DI can be defined as the difference between maximum (PT_{Max}) and minimum (PT_{Min}) execution time to the average (PT_{Avg}) execution time of a task among all VMs. It is given by the formula as below.

$$DI = \frac{PT_{Max} - PT_{Min}}{PT_{Avg}} \qquad (8.2)$$

The aim is to minimize the shifting of already assigned tasks i.e. to reduce imbalance in the cloud eco system. The proposed model is simulated using SLA aware scheduling and load balancing with the aid of prediction mechanism. The detailed explanation of proposed technique is given in section 8.4.

8.4 SLA Aware Scheduling and Load Balancing

Optimal load balancing is one of the main issues in cloud environment. Efficient resource allocation and scheduling will avoid a situation where, some of the hosts are overloaded while; others are idle or engaged with a little work. An efficient SLA aware resource allocation strategy will improve the overall performance of the system that might increase the customer satisfaction.

Since many hosts are present in a datacenter, characteristics of the datacenter are the characteristics of hosts. Hosts have specific processing elements (PEs), RAM and bandwidth characteristics. Each host is virtualized into number of VMs.

VMs in a cloud environment have some specific characteristics like bandwidth, RAM capacity, number of PEs and MIPS rate as like PMs. Characteristics of each VM will differ from another. When the user requests arrive at the broker, the broker will submit it to the VMs at a datacenter for execution. In this proposed architecture before submitting a task into a specific VM, the broker checks the SLA requirement of each task. When these properties are matched with the properties of a particular VM, the task is then submitted to that specific VM for execution. The SLA verification is performed during the initial allocation stage and in the load balancing stage.

8.4.1 SLA verification

SLA is an agreement between the service user and the service provider in a system. In this step SLA requirement of each task is verified according to the properties of suitable VMs. The VM MIPS rate, VM bandwidth and RAM capacity are the various SLA parameters considered in this SLA aware load balancing algorithm. The requirement of each task may vary at different times.

SLA verification is performed during the initial allocation of a task to a VM and also during the load balancing state. User can specify their SLA requirements and, if the VM properties meet the SLA, then user can execute that task on the VM. Whenever a task is going to allocate to a VM, the VM properties like its communication bandwidth, RAM capacity and MIPS rates are checked. The current PM load is also considered for VM allocation, since each task has specific SLA requirements.

To avoid SLA breaches we have calculated the probability of SLA violations using a prediction model. The procedure for calculating the probability of occurring SLA violations are explained in section 8.4.5. Using this prediction model it avoids probable SLA breaches. Thus it achieves the objective mentioned in the mathematical equation (8.1). Even after this SLA violation free allocation, the method checks for violations in each condition (*k*) in SLA during execution of a task. For this the proposed method monitors extensions happened in makespan and parameters related to VMs regularly and apply penalty $\dot{P}_{ij}(k)$ mentioned in the agreement to enforce SLA requirement. Ultimately this helps to reduce penalty and cost of computation.

8.4.2 Load balancing decision

Based on the values of load and standard deviation, the system will decide whether to do load balancing or not. In this module, first check whether the system has the capacity to perform load balancing. Load balancing is only possible when the capacity of physical server is greater than the current load. If the current load of the datacenter is greater than the current capacity, then load balance becomes impossible. This is because; the datacenter is not in a normal condition and hence, the system is not capable for load balancing and scaling of resources will be required for violation free operations. Scaling is the ability of a cloud datacenter to handle growing or decreasing demands; thereby it supports the elastic resource provisioning.

Load balancing is only possible when the load of datacenter is less than its capacity. In this case the load balancing decision is taken on the basis of the standard deviation value calculated using equation (8.3). Here a threshold value is set, and this value is compared with the calculated value of the standard deviation. Up to that threshold value, PM is in normal condition and there is no extra load on that PM. If the standard deviation value is greater than the threshold value then load balancing is needed, because this overloaded PM have some difficulties in handling all these tasks. Therefore, some tasks are to be transferred to other PMs for execution.

Load balancing decision is made using the value of standard deviation (σ). Standard Deviation of load in PM is calculated using the equation (8.3)

$$\sigma_{PM} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_{VM} - \mu_{PM})^2} \qquad (8.3)$$

Where PT_{VM} is the sum of processing time of all the active VMs in the datacenter and μ_{PM} is the average processing capacity of the PM.

8.4.3 PM grouping

The PMs are grouped into overloaded and under loaded PM based on the standard deviation value of load and the threshold. Each group contains a set of PMs. Task withdrawn from one of the overloaded PM set has to be assigned to an under loaded PM based on the load, SLA parameters and tasks already assigned to the particular under loaded PM. In this method, PMs whose standard deviation values greater than threshold are considered as the overloaded PMs and the PMs whose standard deviation values less than threshold is considered as under loaded PMs. Since threshold and load on each PM is changing at every minute, the overloaded and under loaded PM list is getting updated. Tasks removed from overloaded PMs are assigned to the under loaded PMs for execution. The task is executed only if, the under loaded PM set contains VM having desired properties.

8.4.4 Task transfer

In the PM grouping module, PMs are categorized into overloaded and under loaded PM based on the threshold mentioned in [172]. Since the load balancing is SLA aware, before checking the SLA, it’s necessary to find the demand of overloaded PM and the supply of under loaded PM. Here, demand means requirement of overloaded PM. Supply means the availability of the under loaded PM. Demand (resource request to a PM) and supply (resource allocation) can be calculated as follows:

Resource allocation to an under loaded PM set is given by:

$$Supply = \ Maximum\ PM\ Capacity - \frac{L_{PM}}{C_{PM}} \qquad (8.4)$$

Demand of each machine in overloaded PM set is:

$$Demand = \frac{L_{PM}}{C_{PM}} - MaximumPMCapacity \qquad (8.5)$$

In order to perform load balancing, the system needs to identify the demand of each overload PM and supply to the under loaded PMs. The task transfer occurs only when the demand meets the supply. The tasks which were withdrawn are allocated to the PM with highest capacity. Only based on the priority, the task which is to be transferred or migrated is selected from the overloaded PM. The task which has least priority is selected for migration. Since, they have not started execution; it will be easier to migrate. This migrating task also comes with some SLA requirements. The task is allocated to a PM among the under loaded PM set, which has the desired SLA characteristics under current load situations.

8.4.5 SLA violation detection and VM scaling

In the proposed method initially the VMs are created with minimum specification such as memory, MIPS rate, and VM bandwidth. When a particular VM consumes less processing power and memory than reserved, the remaining memory, processing power and the VM bandwidth are collected into a resource pool. The resource from this consolidated resource pool can be shared to VMs, that require more processing power, memory, etc. The procedure is shown in the figure 8.2.

SLA violation occurs when a VM fails to meet the requirement of a task such as CPU speed, RAM, and bandwidth. If large number of requests arrives to the same PM, and if it has to serve all these requests, then scaling is performed to reduce SLA violations. VM bandwidth, VM MIPS rate and VM RAM capacity is scaled to a particular amount.

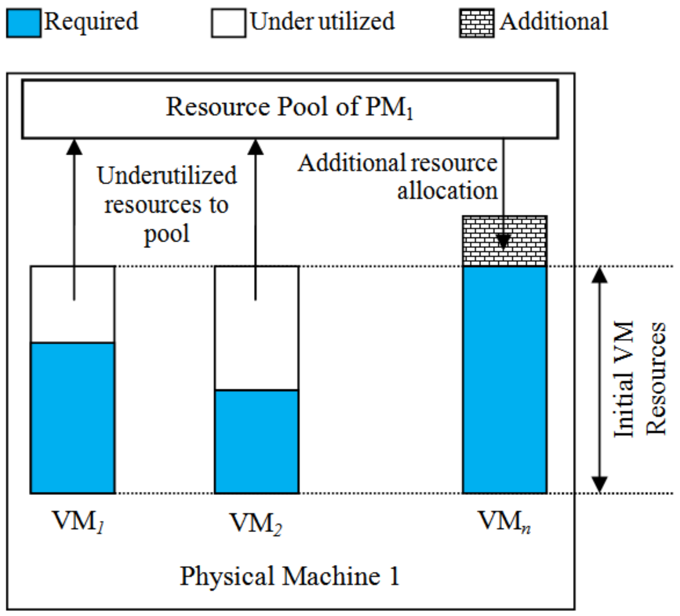


Fig. 8.2 Underutilized reserved VM resources are collected in the PMs resource pool.

For each new task SLA aware load balancing algorithm is shown in figure 8.3 and respective enhanced resource allocation policy is shown in figure 8.4.

Algorithm: SLA aware load balancing

```
Start
For each task verify the MIPS rate, bandwidth and RAM capacity of PM and VM and allocate task.
    Calculate the load and capacity of each PM.
    Group the PMs based on load as overloaded or under loaded based on the standard deviation value in equation (8.3) and threshold value T.
    Find the supply of under loaded PMs and demand of overloaded PMs based on equations (8.4) and (8.5).
    Sort the overloaded and under loaded PM sets based on load
    Sort the tasks in overloaded PMs based on priority.
    Select the least priority task for migration to the under loaded PM.
    Find the capacity of PMs in the under loaded set.
    For each task in each overloaded PM find a suitable under loaded VM in PM based on capacity and SLA requirement.
    Update the overloaded and under loaded PM sets.
Stop.
```

Fig 8.3 SLA aware load balancing algorithm

8.4.6 Probability of SLA violation and penalty

The folded normal distribution [232] measure the probability of the normal distribution on $(-\infty, 0]$ is folded over to $[0, \infty)$. It is a distribution of the absolute value of a random variable with a normal distribution. In dynamic cloud resource allocation problem the main focus is on the magnitude of incoming customer requests, which is a normally distributed variable, then the folded normal distribution is a natural solution to calculate probability.

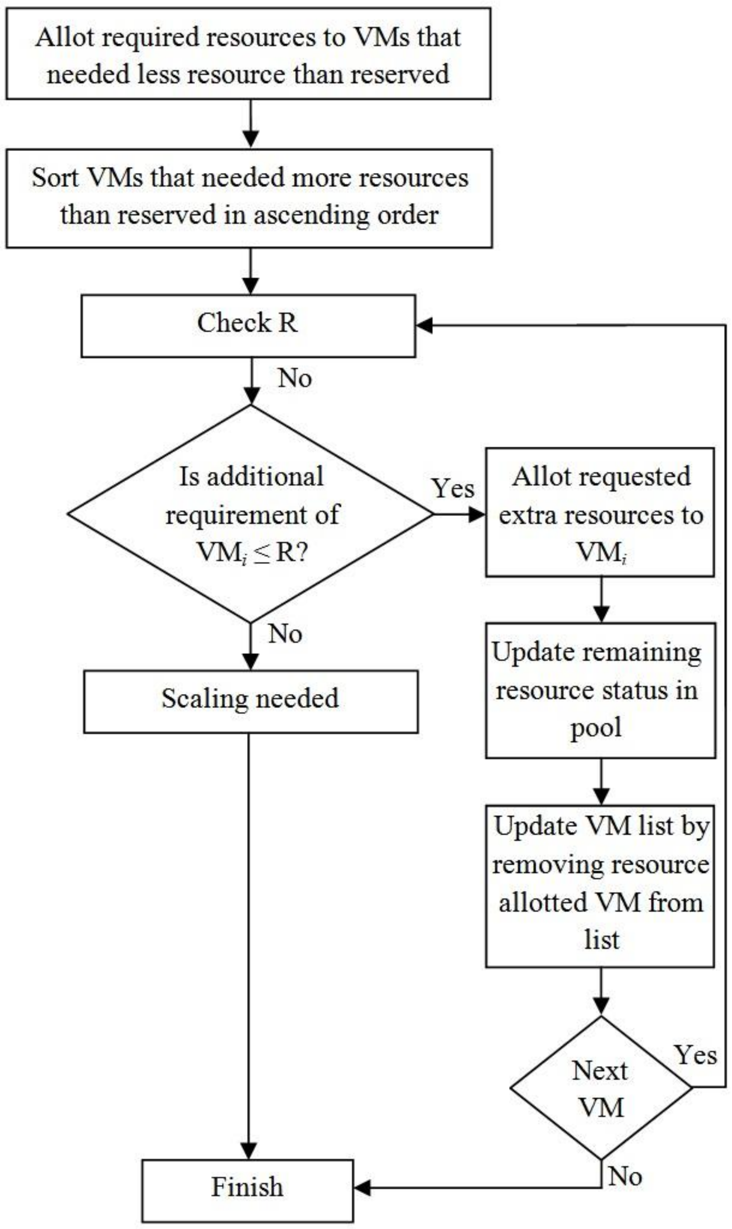

Allot required resources to VMs that needed less resource than reserved
Sort VMs that needed more resources than reserved in ascending order
Check R
No
Is additional requirement of VM$_i$ ≤ R?
Yes
Allot requested extra resources to VM$_i$
No
Scaling needed
Update remaining resource status in pool
Update VM list by removing resource allotted VM from list
Next VM
Yes
No
Finish

Fig. 8.4 Enhanced resource allocation policy

The probability of occurrence of a SLA violation is high when the aggregate resource requirements of all VMs executed on a PM is greater than the maximum capacity of resources or processing power available on a PM:

$$\mathrm{P}\begin{bmatrix}\text{SLA violations on}\\ \text{a particular PM}\end{bmatrix} = \mathrm{P}\left[\sum_{k=1}^{n_i} Requirement > \text{Capacity}_{\text{PM}}\right]$$

Since the input load follows folded normal distribution it can be written as Y $\sim N(\mu_{\text{PM}}, \sigma_{\text{PM}}{}^{2})$

Here $\mu_{\text{PM}} = \sum_{\text{i}=0}^{n_i} \mu_{\text{i}}$ and $\sigma_{\text{PM}} = \sqrt{\sum_{\text{i}=0}^{n_i} \sigma_{\text{i}}^{2}}$

Here μ_i is the average and σ_i is the standard deviation of all the incoming length (in MIPS) of user tasks to VMs in a PM which follows standard normal folded distribution with N (10000, 3500).

The aim is to place user specified VMs on optimal physical servers by considering the SLA and cost. Cost analysis is necessary to allocate a VM to a PM. The computation cost associated with each PM is calculated for each requested VM_i based on the equation (8.6). While allocating VM to a particular PM, the mechanism should consider the objective of the service provider as well as the customer. Here the service providers have to serve several customer requests simultaneously, so they are trying to reduce the SLA violations, while maximizing the profit. By considering these factors the algorithm places VM to the best available active PMs.

Cost $PM_i = \alpha.\ P[PM_i \geq 1/VM_i \epsilon PM_i]$ +

β . P[Next user rejected by $PM_i/VM_i \epsilon PM_i$] (8.6)

Where $\alpha, \beta \geq 0$.

Here the probability of rejection (β) depends on server processing power, number of simultaneous request from the users, network traffic, etc., so that the situation is dynamic and algorithm has no control over these parameters.

When allocating resources, possibility of the rejection of a service depends on unreserved resources in the pool. Let S is the total processing power (in MIPS) available in a PM, then likelihood of rejection of next VM request can be calculated as

$$\text{Probability of Rejection of next user} = \left[1 - \frac{\sum_{VM_i \epsilon PM_i} \mathrm{t}_i}{\text{Capacity}_{\text{PM}}}\right]$$

So that the cost function in equation (6) becomes

$$\text{Cost PM}_i = \alpha.\int_S^\infty N(\mu_{\text{PM}}, \sigma_{\text{PM}}{}^2) + \beta.\left[1 - \frac{\sum_{VM_i \epsilon PM_i} \mathrm{t}_i}{S}\right] \qquad (8.7)$$

The scaling is determined based on the probability of SLA violations. The aim of this method is to reduce number of SLA violations with the help of a prediction model. The prediction model works on the basis of the past usage pattern, which helps to find out the right quantity of resources required. Here the past usage pattern is simulated based on the Lublin model [223]. The scaling process is carried out using this prediction to avoid SLA breaches. Figure 8.5 shows the process of enhanced resource allocation policy with scaling process initiated by the prediction mechanism.

8.4.7 Significance of Alpha and Beta

The performance of the proposed algorithm depends on the SLA violations and likelihood on SLA breaches so that VM placement depends on this probability.

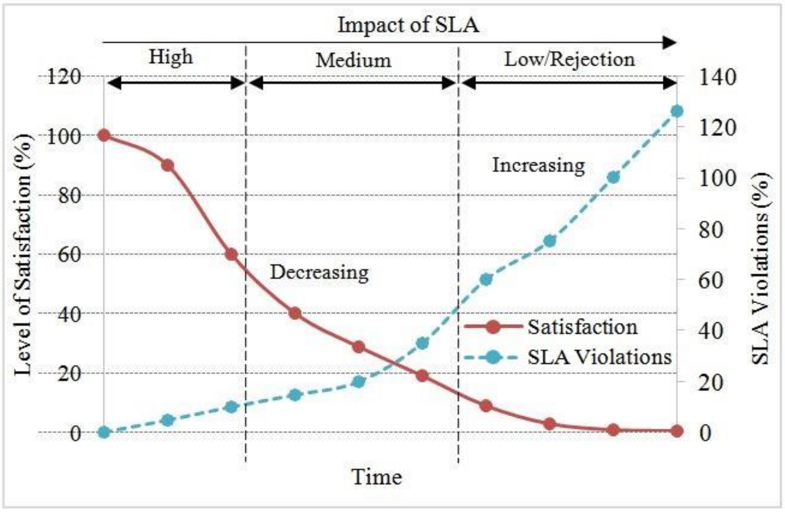


Fig. 8.5 Significance of α and β

In the above equation (8.7), the significance of alpha and beta cost of SLA breaches (α) and cost of service rejection (β) is shown in the figure 8.5. If the cost factor dominates then algorithm tries to allocate maximum number of VMs to a PM which increases the chances for SLA breaches. By setting suitable α and β value, providers can adjust these values depending on cost of SLA breaches and service rejection cost. By adjusting the ratio, the provider can optimize their revenue by minimizing SLA breaches and service rejection.

8.5 Experimental Setup and Results

The tasks arrive at random time interval to the cloud providers. So to test the scalability of the algorithm static workload is inefficient, because the elastic cloud uses pay-as-you-use model. In order to make the realistic environment we have used two workload generation model for user request generation; Lublin model [223] and Synthetic Random [234] workload.

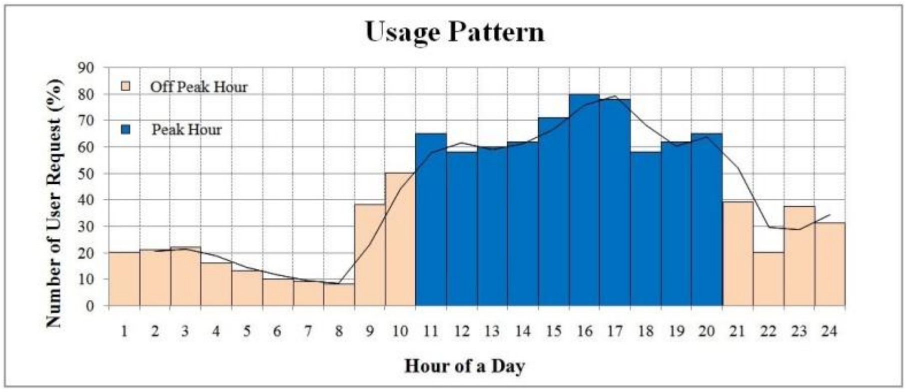


Fig. 8.6 Cloud resource usage pattern during a day – Lublin model

The arrival of tasks to the CSP is modeled by Lublin model because this model considers the daily cycle of arrivals depending on the working hours as shown in figure 8.6. This distribution models are based on the Gamma distribution. Gamma distribution is good in modeling probabilities such as queuing analysis and for sets of values that may contain skewed distribution. During morning sessions, there will be lesser number of active users, which gradually increases as the day moves to business hours and load increases from low to high. In this situation the allocation policy selects optimal machines and reserves the resources as user requested. If the convenient VMs are unavailable in the active hosts it will wake up the additional hosts to satisfy the customer needs. In order to maintain the QoS and reduce SLA violations, the optimization and load balancing algorithm reallocate the tasks from heavily loaded machines to the low loaded hosts. By the end of the day the numbers of active users are less. As the users leave the CSP, some hosts are heavily loaded while others are idle. In this stage a server consolidation mechanism is needed to reduce the power consumption. So a server consolidation algorithm is applied and idle hosts are put into sleep mode.

The performance of the SLA aware load balancing algorithm has been evaluated by simulation using CloudSim toolkit [129]. CloudSim-3.0 is used as a framework in the simulator environment. Modeling and simulation of large scale cloud computing data centers, hosts and virtual machines are provided by CloudSim simulator. The main components in CloudSim are datacenter, virtual machine and cloudlet. The parameters for the simulation environment is shown table 8.2.

Table 8.2: Parameters for simulation environment

Parameter	**Value**
Number of tasks in peak hour	Upto 2000
Number of tasks in off peak hour	Upto 5000
Physical Machines	100
Virtual Machines	1000
Threshold [29]	60%
Cost	$1-$3

Physical server or host’s utilization is defined as the percentage of time the CPU is busy. It is also referred to as the percentage of the CPU's capacity that is currently being used. In any server machine, when CPU utilization exceeds a certain threshold value, thrashing sets in. Usually to avoid critical conditions in most datacenters, each host has a maximum threshold (normally 60%) [172]. This threshold value indicates that if more than 60 percent of the server machine's capacity is used, an SLA violation is flagged. As in the physical host the threshold fixed for VM CPU utilization is also fixed as 60%. CPU utilization can be measured from hypervisor and this 60% mark will ensure better response time for other applications running on VMs hosted on the same hosts. As most customers have expectations on faster application response time, even slight increase in response time (above the predefined threshold) can result in SLA violations.

VM failures can bring in SLA violations; they can also be caused by progressive performance degradation of the application which occurs due to software failures or high workload conditions. The degradation usually results in an increase of server machine CPU consumption, virtual machine utilization, delay, application response time and VM migration time. Consequently, this may lead to an imbalance in the cloud eco system.

8.5.1 Impact of workload on scalability

The scalability of the proposed algorithm is tested with 100 PMs with maximum 10 VMs per PM. The experimental results are shown in the figure 8.7 (a) and (b).

From figure 8.7 (a) it is clear that the average number of migrations goes up linearly for the increase in number of tasks. Here average number of migrations is 0.908 and 0.84 for Lublin model for peak and off peak hours respectively. For random workload this is 1.62 and 1.226 for peak and off peak hours. The number of migrations for random workload is higher than the Lublin in peak hours. This is because; to maintain SLA the system has to use more PMs in active state due to random load.

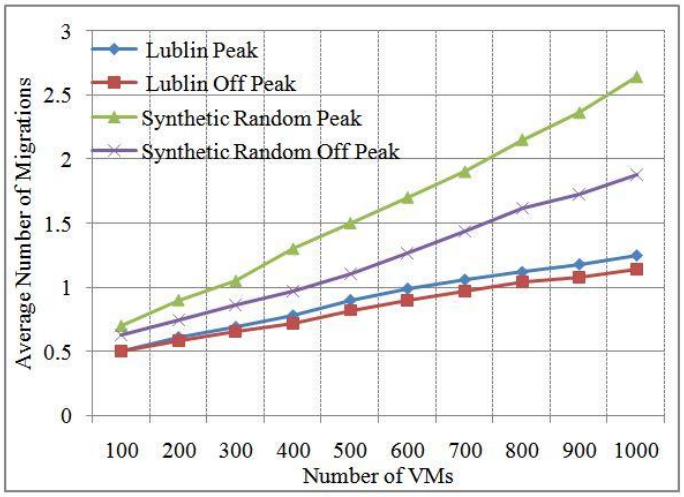


Fig. 8.7 (a) Average number of migrations

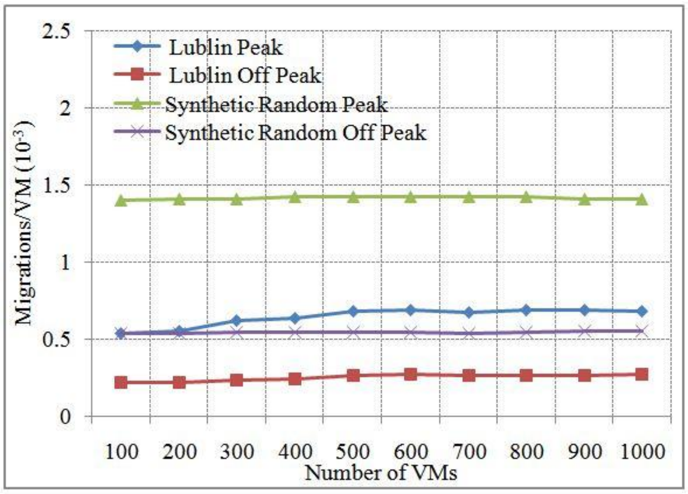


Fig. 8.7 (b) Average number of migrations per VM

Figure 8.7 (b) contains average number of migrations per VM. Here the migrations are almost linear in nature. There is only a small increment in number of migrations per 1000 VM. This justifies the previous results in figure 8.7 (a). These results show that algorithm is capable for large scale operations in real cloud environment.

8.5.2 Load prediction

The efficiency of a distributed cloud system can be improved by right prediction about how much resources are required and time duration. Prediction mechanism allows the scheduler to allocate computing resources based on the customer requirement in time. With the aid of this prediction model, the proposed method increases the workload handling capacity, i.e., service providers can effectively use their computing resources or they can scale-down or scale-in their capacity. This increases resource utilization rate.

The optimal resource prediction is shown in figure 8.8.

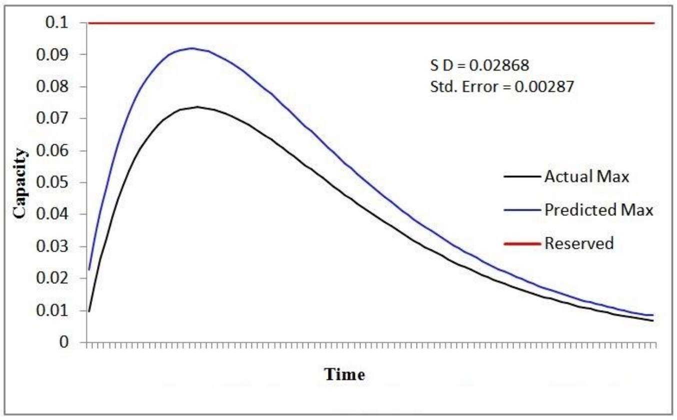


Fig. 8.8 Optimal resource prediction

If the scheduler can predict or assess the requirement for next time period based on the past requirement, then it can allot those resources to the next customer task in the queue. If the resources are not enough to satisfy the customers’ requests, it can go for resource scaling or elasticity across the different datacenters in the cloud environment. The optimal prediction mechanism will reduce the number of VM migrations or consolidations. Figure 8.8 shows the empirical load prediction using proposed Lublin model over time. The standard error is one of the best methods in measuring the standard deviation of a sampling distribution. The standard error in predicting the optimal load for the proposed method is only 0.00287.

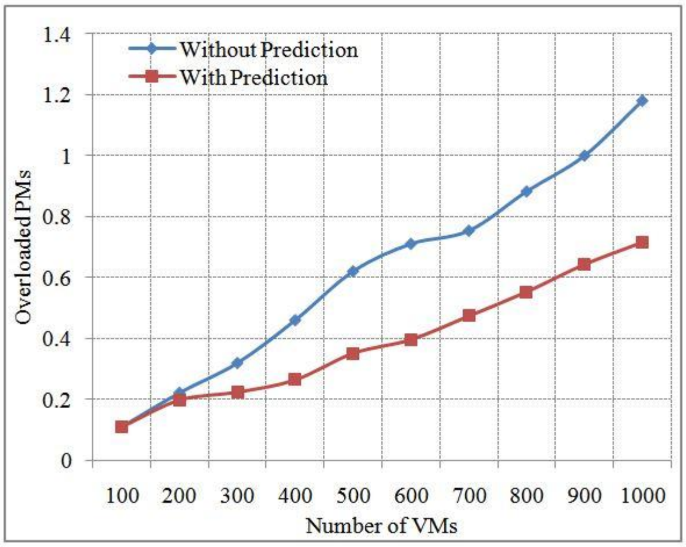


Fig. 8.9 (a) Overloaded PMs

The figure 8.9 (a) shows the number of overloaded PMs during the user task execution with and without prediction mechanism. The prediction reduced the average percentage of overloaded PMs from 62% to 39% in the simulation environment with 100 PMs. This 23% reduction will improve the performance of the provider to maintain the SLAs. Again, the average number of migration per VM is measured with and without load prediction. In the figure 8.9 (b) the proposed mechanism experiences 14.3% lesser migrations due the prediction mechanism than without prediction. This will increase system stability due to lesser number of VM migrations. This result is verified with system imbalance analysis, which is given in figure 8.12.

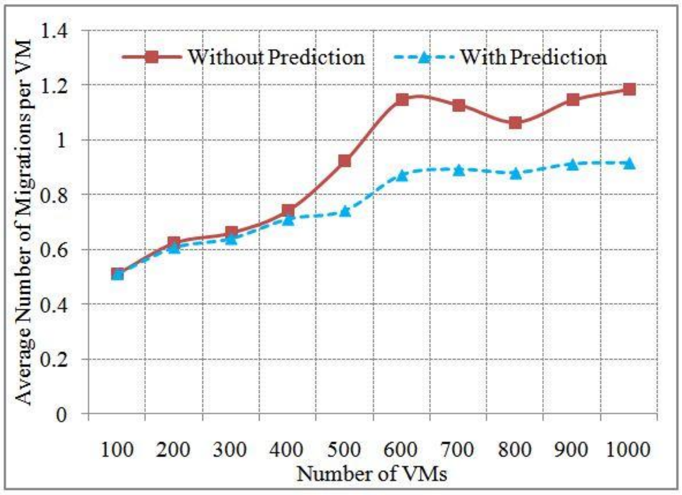


Fig. 8.9 (b) Number of migration with load prediction

8.5.3 Makespan

Makespan is the overall task completion time. It is the difference between intial task submission time and its completion time. Figure 8.10 and table 8.3 show the makespan comparison of RR [223], ABC [98], Random Power Aware [220], Max-Min [194] and SLA aware load balancing in peak hours mentioned in figure 8.6. From the figure 8.10 and table 8.3, it is clear that makespan is reduced to a considerable amount by using SLA aware load balancing.

Table 8.3: Makespan comparison (Milli Seconds)

Number of Tasks	RR [223]	Random Power Aware [220]	Max-Min [194]	ABC [98]	SLA Aware
100	403.83	387.86	385.14	380.10	375.41
200	791.22	765.41	761.55	750.10	721.92
300	1278.61	1242.96	1197.96	1121.18	1065.43
400	1672.60	1600.51	1614.37	1530.54	1414.94
500	2093.39	1998.06	1950.78	1867.15	1770.45
600	2459.78	2275.61	2297.19	2235.66	2007.96
700	2879.17	2673.16	2703.60	2580.75	2424.47
800	3285.56	3065.71	3090.01	2930.31	2850.98
900	3772.95	3508.26	3456.42	3331.92	3157.49
1000	4390.34	4085.81	3972.83	3725.36	3490.00

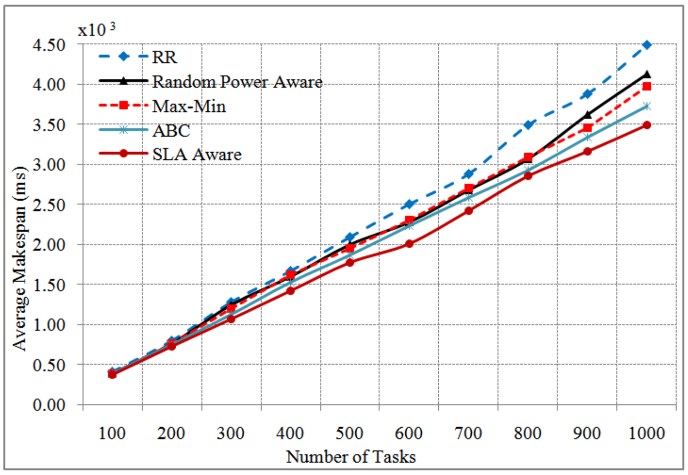
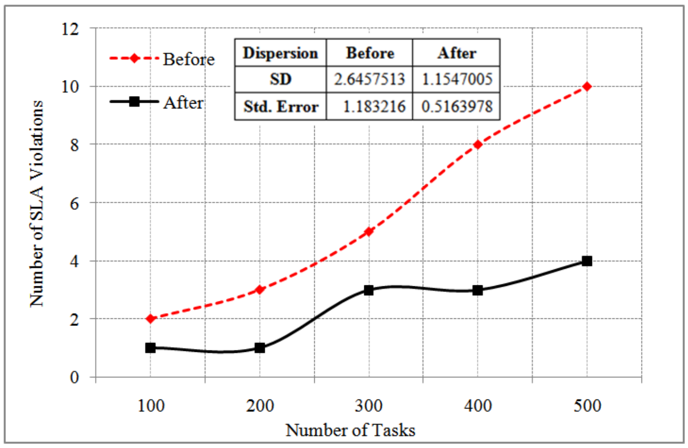


Fig. 8.10 Average makespan

8.5.4 SLA violations

Figure 8.11 (a) shows the number of SLA violations before and after scaling. From the figure it is clear that scaling reduces the number of SLA violations than other methods. The standard error in predicting the scaling is reduced from 1.183 to 0.516, which shows the accuracy of the method.



Dispersion	Before	After
SD	2.6457513	1.1547005
Std. Error	1.183216	0.5163978

Fig. 8.11 (a) Average number of SLA violations before and after scaling

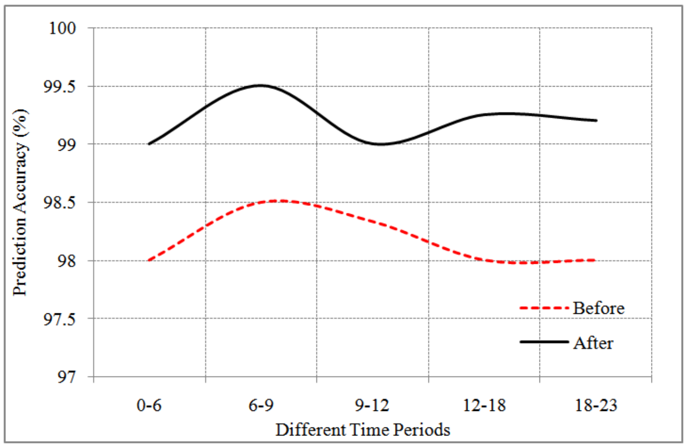


Fig. 8.11 (b) Prediction accuracy

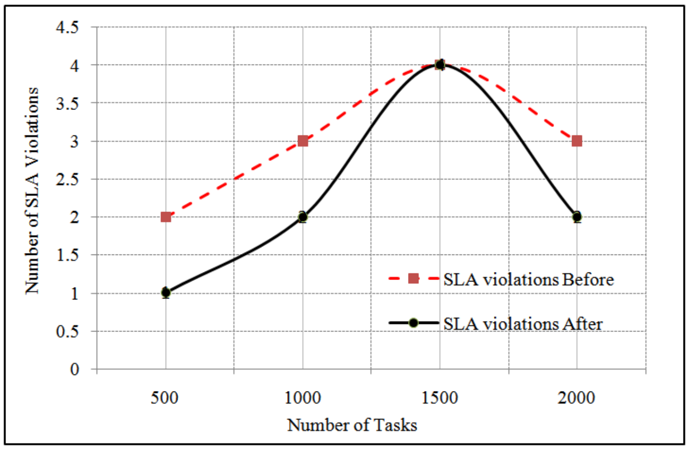


Fig. 8.11 (c) Average SLA violations during off peak hours

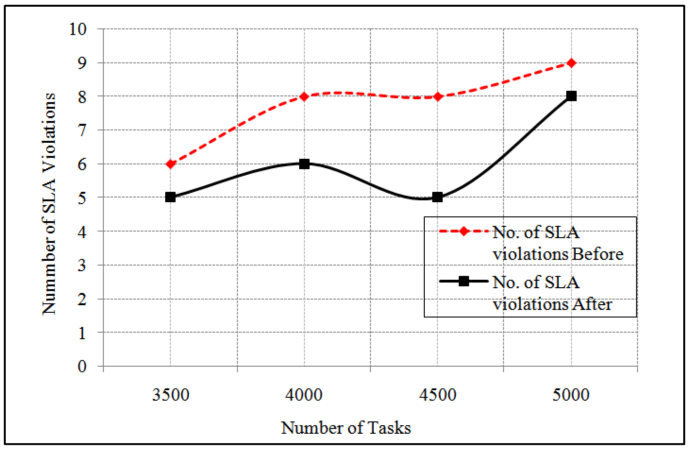


Fig. 8.11 (d) Average SLA violations in peak hours

The proposed method predicts the SLA violations on peak and off peak time. The prediction accuracy is measured and is shown in the figure 8.11 (b). For measuring the accuracy the entire day is divided into 5 sessions depending on the task arrival intensity, which is based on the prediction model shown in figure 8.6. The extensive simulation results along with the different experimental settings showed an overall prediction accuracy upto 99.5%.

Here also the number of tasks on off peak hours is considered in the range 500 to 2000 and on peak hours up to 5000. Figure 8.11 (c) and 8.11 (d) shows the number of SLA violations on off peak hours and peak hours respectively. In both cases, the method is able to reduce the number of SLA violations.

The above results shows that the proposed approach reduced number of SLA violations as we have modeled in the mathematical equation (8.1) with the aid of prediction model. Penalty enforcement also forces service providers to keep the conditions in SLA during the execution of a customer tasks.

8.5.5 Imbalance

The frequent migrations in the cloud causes load imbalance, which adversely affect the performance and reduce QoS delivered to the customers. The proposed method reduces the number of migrations (DI factor) both in the initial resource allocation stage and load balancing stage. As in the mathematical model, the proposed method reduces the number of migrations (DI factor) both in the initial resource allocation stage and load balancing stage. This is because of the proper prediction that avoids situation of frequent migrations and thereby related potential impact on makespan.

From figure 8.12 the DI is compared with Max-Min, RR, ACO [89] and modified throttled [131] algorithms for different number of tasks. The average DI factors are 3.71, 3.61, 2.65 and 3.42 respectively for Max-Min, RR, ACO and modified throttled algorithms, while the proposed method have only 1.63. Hence it is clear that the proposed method reduces the imbalance to a substantial amount and thus the reduction in imbalance results in better QoS for customers.

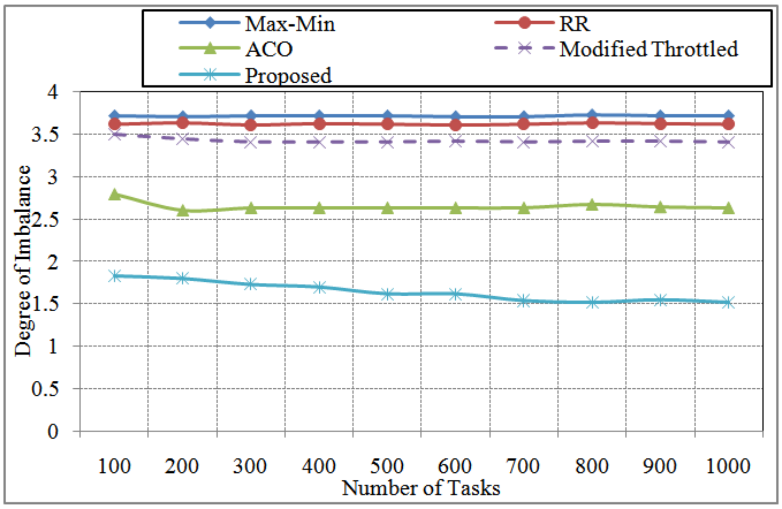


Fig. 8.12 Degree of imbalance using SLA aware load balancing comparison with Max-Min, RR, ACO and Modified Throttled algorithms

8.5.6 Cost

To study the effect of the increase in workload, the experiments were conducted for varying number of input tasks. The result shows that, the proposed method incurs lesser cost than the non-SLA aware method. When more number of tasks is entering into the cloud, the cost of computation also increases as shown in the figure 8.13. This is because, in SLA aware method, the cloud broker considers the current status of PMs and distributes the tasks by evaluating the conditions based on the SLA requirements. While in non-SLA method, the user requirements are never considered for resource allocation.

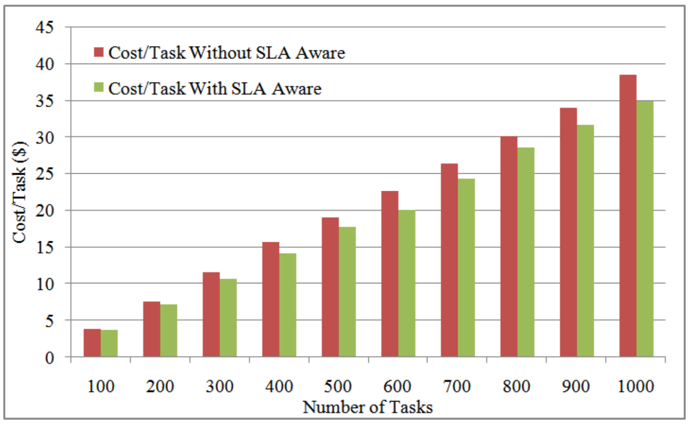


Fig. 8.13 Cost benefit analysis for SLA aware method

All the above results show that the proposed SLA aware load balancing and scheduling algorithm reduces the makespan, degree of imbalance and number of SLA violations in the cloud environment, which give better performance to the end users in terms of time and cost, with very less SLA violations. This is achieved with the help of optimal allocation with prediction methods and enforcement of SLA with penalty and auto scaling. The method efficiently uses the cloud resources.

8.6 Summary

This chapter proposed an integrated quality assured SLA aware load balancing and scheduling algorithm for the cloud environment. This algorithm migrates tasks from VMs in overloaded hosts and submit it to the VMs in the under loaded hosts having highest capacity. This algorithm considers VM processing power, VM memory capacity and bandwidth as the SLA parameters. During the initial allocation and load balancing stage, a task is submitted to a VM that meet users' SLA requirements. The experimental results proved that, the proposed integrated SLA aware load balancing and scheduling algorithm have minimum makespan compared to Random, RR, Min-Max and ABC algorithms. It also reduced frequent migrations i.e., degree of imbalance into a considerable amount. It is cost effective and SLA violations are reduced using proper prediction method with timely scaling algorithm thus the proposed method ensure QoS in cloud scheduling.

Chapter 9
CONCLUSIONS AND CONTRIBUTIONS

Contents

9.1 Overview

The theme of the thesis is centered on the quality of service in cloud scheduling. Cloud computing is an innovative computing paradigm designed to provide flexible and low cost way to deliver IT services on-demand over the internet. Proper scheduling and load balancing of the resources are required for efficient operations in the distributed cloud environment. Since cloud computing is growing rapidly and customers are demanding better performance and more services, scheduling of the cloud resources that guarantees Quality of Service (QoS) have become a very interesting and important area of research. Hence developing scheduling policies that confine with the user's practical needs and constraints would be extremely useful in cloud virtual machine systems. Makespan, cost, efficient load balancing with stability, scalability, and energy consumption are important factors for providing good QoS in the cloud resource allocation process. Also, the scheduling policy will be beneficial to both service providers as well as customers. It should also allocate adequate resources for the best performance of user applications and to meet service level agreements while considering the energy efficiency of a cloud data center. Considering these factors we have designed and developed QoS oriented scheduling policies that will consider minimization of makespan, cost, energy consumption and SLA violations with improved stability and scalability in this thesis. The experimental results proved the efficiency of the proposed methods.

The major findings in this thesis are described below. We have done a comprehensive survey about various scheduling methods proposed for cloud and identified shortfalls and need for improvement in achieving QoS. In the first method we have improved the QoS through the reduction in makespan by an efficient VM placement method. The second method handled makespan-migrations to improve QoS. From these we can conclude that quality in the cloud can be improved with efficient makespan-migrations methods.

Our next finding is that active physical machine clustering improves the energy efficiency of the data centers. Since clustering improves resource utilization, unused or idle physical servers can be switched off and they can be reintroduced when the workload increases, thus improving energy efficiency.

When the workload increases in a physical server there will be performance interference due to the sharing of common resources. We have modeled a mathematical equation for the total load on a system considering the parasitic load due to interference. Based on this, a regression model is developed to achieve QoS in the cloud by controlling frequent migrations. Thus this method improved the stability in the cloud.

SLA enforcement can be done through auto scaling mechanism in the cloud. For this, we have used the principle of locality property of Petri Net for effective scaling decisions to achieve QoS. Another finding of this thesis is the controlling of SLA violations through the enforcement of penalty and the use of a workload prediction mechanism.

9.2 Research Contributions

Designed and developed QoS guaranteed scheduling mechanism for cloud. The research contribution of the thesis consists of:

- Makespan is one of the important parameters in achieving QoS in the cloud. We have developed VM placement scheme to handle makespan. This scheme also minimizes the storage requirement as well as power consumption.
- Cloud scheduling is an NP-hard problem. Hence, intelligent methods are needed to arrive at near optimal solutions to mitigate the issues related to the dynamic nature of cloud resources. We have successfully developed and tested hybrid method based on an evolutionary algorithm for VM-migration through load balancing. This method minimized makespan and imbalance in the cloud ecosystem.
- Cloud server farms consume huge energy. Some of the machines may be in an over loaded or under loaded stage. For energy efficiency, better energy management policies are needed. In order to address this energy concern, this thesis

contributed an energy efficient clustered load balancing mechanism for server farms promoting green computing. It improved energy efficiency through active physical server clustering based load balancing.

- In physical cloud, the total computation power of a physical machine cannot be used due to some interference created by the sharing of common resources. It also results in creating a parasitic load on the system. We have developed a novel interference aware prediction model to enhance the stability in the cloud ecosystem. This mechanism reduced the performance interference in the cloud datacenter with the aid of an optimal prediction mechanism. This mechanism improved the performance of the service provider by predicting optimal threshold range for the maximum efficiency for physical servers.
- Maintaining conditions in the SLA is a major step in achieving QoS in the cloud. We have developed an SLA enforcement mechanism with auto scaling. This dynamic provisioning system with scaling policy reduced makespan, number of SLA violations, penalty cost and maximizes profit. The development of a Petri Net model for the cloud to enhance QoS is another contribution of this thesis.
- The methods proposed in this thesis also address the load balancing and reduced imbalance due to frequent migrations happening in the cloud. We have successfully developed and tested the models that reduce frequent migrations thereby achieving better load balancing and increased stability in the cloud datacenters.

- Finally, developed an integrated SLA enforcement scheme that will consider makespan, migrations, SLA and cost with the aid of a prediction model. The incorporated prediction model is based on the past usage pattern and forecasts future SLA violations due to fluctuating workload. Based on these forecasts appropriate load balancing and scaling decisions are carried out, which reduced cost, makespan and SLA violations. This method also improved system stability due to the scaling mechanism by limiting frequent migrations. All our contributions mentioned above resulted in better QoS delivery in the cloud.

9.3 Proposals Made in this Thesis

In this section, we highlight how different chapter's progress to accomplish different objectives of the thesis and the difference between the outcomes of each proposal made in the thesis.

The proposed works mainly deal with the enhancement of cloud scheduling process to improve the quality of service. Makespan is an important factor in achieving quality in the cloud environment. The first method proposed is to enhance the makespan with the maximization of available resources. For handling makespan it effectively used Best-fit – Remaining-fit strategy. It also capable to minimizes the storage overhead. The experiments have proven that this method is capable to maintain QoS.

The second method proposed is based on load balancing which handles makespan through migrations. Since makespan is an incredible parameter for QoS satisfaction, considering it with migration strategy produces improvement in the scheduling process. Here, the power of swarm intelligence is used to reduce makespan and VM migrations thus it achieved quality. The experimental result obtained proved that the proposed scheme attained the objective to enhance the makespan.

Cloud datacenters contain several physical servers that consume a huge amount of electricity. Reducing energy usage is a good move toward green computing. The third technique is a physical sever clustering mechanism for improving energy efficiency. In this load balancing technique, the number of active machines can be reduced. If the currently active servers not enough to meet the QoS requested by the customer, then only it considers an inactive idle servers. The clustering of active physical machines and energy aware virtual machine migration reduces energy consumption.

The fourth method is to enhance system stability through an interference aware mechanism. If we have developed a proper prediction mechanism to know the optimum load that can be processed at a server, the overall system performance can be improved. i.e., if we know the optimum workload that can be processed at a server based on the currently available processing power, the system can avoid frequent virtual machine migrations. Thus the system achieves stability by limiting frequent migrations and respective performance degradation. Our proposed method is capable to achieve system stability through the prediction mechanism. The prediction gives a prior knowledge about the workload to be handled by a server. For this model, we have formulated a novel regression model for parasitic load due to interference. The real implementation and obtained results proved that the proposed mechanism accomplished the objective.

The fifth mechanism deals with SLA enforcement with auto scaling mechanism. Here auto scaling of resources is adopted to avoid violations in the SLA conditions. To enforce SLA, the penalty is applied in the case of any breaches in agreed conditions. If the penalty is imposed for violations in the SLA conditions, then service providers are keen in maintaining the agreed QoS. To assist auto scaling process a Petri Net model for cloud is also designed in this method. We have employed principle of locality to improve the auto scaling process. Auto scaling again reduces the migrations, thus it enhances the system stability factor. The timely scaling mechanism helps to reduce SLA breaches so that profit of the service provider can be increased. Thus our proposed scheme adheres with the objective of the thesis.

The sixth technique again addresses the SLA enforcement, with an integrated approach to achieve good quality of service. This method considers makespan, SLA, cost with penalty, scalability and stability. Here the financial obligations due to SLA violations are calculated before for making scaling decisions. The probability of SLA violations and penalty is calculated for this purpose. The impact of workload on scalability is also a factor to maintain QoS. So an optimal scheduling mechanism with load, system stability and cost is designed in this technique to cope with SLA and cost.

The fundamental goal of the thesis was to design and development of QoS guaranteed cloud scheduling techniques with performance improvement in terms of makespan, energy efficiency, stability, SLA enforcement, and cost. In this thesis, a progressive approach was followed to accomplish this objective.

9.4 Performance Study

In this section we have highlighted a detailed performance analysis of the proposals made in the thesis.

Chapter 2 presented a comprehensive review in the area of resource allocation in cloud computing. This review has offered promising changes in this area. It identified highlights and limitations of different methods for resource allocation including scheduling, load balancing and VM placement. The analysis of current literatures has assisted in finding gaps and identifying research challenges that have clarified the direction of this thesis. Chapter 2 also identified the metrics to evaluate the performance of the system.

The Chapter 3 proposed Bin packing based algorithm to minimize makespan and maximize resource utilization in a cloud datacenter. It also focused on the profit of a cloud service provider. It proposed a Best-fit – Remaining-fit strategy that efficiently places the virtual machines to minimum number of active physical servers. The jobs are scheduled using Best-fit approach. The cloud broker employs Remaining-fit method for VM placement.

Here we have considered each PMs in the datacenter are bins. The VMs requested by the customer, are the objects; which are to be filled in the bin. Our algorithm attempts to minimize the number of PMs required for placing customer requested VMs. At the same time the algorithm also aims to reduce makespan. This method consists of two phases. In the first phase the jobs are submitted to the cloud through a cloud broker using Best-fit method. In this step, the jobs are sorted in ascending order depending on the processing power required. The available VMs are sorted in a list based on their processing power. Then the cloud broker places the jobs in the job queue to available VMs. By the repeated simulated study we have proved that the Best-fit – Worst-fit approach efficiently maps VMs to the active PMs, such that makespan, power consumption and thereby computation cost are minimized. Thus it’s a promising method to achieve better QoS in the cloud.

In chapter 4 we have proposed and experimented a bee colony algorithm for makespan improvement through efficient load balancing in cloud. In this method, we have used the power of swarm intelligence algorithm to remove the tasks from overloaded servers and migrated these removed tasks to the most appropriate underutilized or under loaded servers. This migration policy also considered priority of the tasks in the waiting queue. The tasks with least priority are selected as candidates for migration. Hence no tasks are needed to wait a longer time in order to get processed and improve customer satisfaction. In this proposed method honey bees foraging behavior to find a food source is mapped into the cloud environment for effective load balancing. Here tasks in the overloaded machines are removed based on the priority. The task with lowest priority is transferred to under loaded resources.

The proposed method works in four stages. In the first stage load on each VM is calculated by adding all the workloads in a PM. In the next stage, load balancing decision is taken based on the load deviation. In the third stage, the VMs are grouped into overloaded and under loaded VMs based on the load on it. In the final step, the tasks are transferred to the under loaded VMs based on priority. Our experimental results proved the efficiency of the proposed approach in terms of low makespan, number of migrations and degree of imbalance.

In order to harness the green energy concept, the importance of improved energy efficiency is proposed in chapter 5. It proposed an energy aware clustered load balancing system in which, heterogeneous cloud resources are grouped into different clusters, by using a partitioning based clustering algorithm. The method progresses through three stages. First, it clusters active physical servers into clusters based on the currently available processing power. Since the search process is carried out only on a particular cluster searching time will be reduced. Then an energy aware VM migration is carried out. Finally, the tasks are assigned to these VMs using process allocation algorithm. This chapter also used a Best-fit – Worst-fit strategy to place the virtual machines to minimum number of active physical servers. Here best-fit VM allocation is carried out based on a weight value of the resource. This weight value depends on its memory, storage and processing capacity of the resource. Then the corresponding VM cluster is found out using this weight value. If suitable resources are available, then allocate it, else goes to second portion of that cluster and check the resource availability. If the VM is unable to allocate in that cluster, then the method checks in other clusters. Finally, again best-fit allocation strategy is used for allocating processes to the VMs. Thus the Best-fit algorithm achieve best VM placement. Here clustering reduced the number of resources needs to be searched and hence reduced the total searching time required for resource discovery and allocation. By the simulated study we have proved that the proposed method reduced energy consumption and thereby computation cost is minimized. The method reduced time for resource discovery, resource allocation and response time with power consumption.

Chapter 6 considered a new parameter called interference for resource allocation in the cloud. It proposed an interference aware prediction mechanism (PiA) for VM migration with auto scaling. Since several VMs with different applications are running in a PM, there will be performance degradation that causes interference in the performance of the system due to sharing of common resources. The proposed work is intended for the stability in the performance and scalability of resources when the user workload increases beyond a certain threshold value. So, VMs in a particular host can be migrated to appropriate destinations based on the least interference values, for the performance improvement of entire cloud system. This will reduce the number of migrations in the cloud system.

The proposed method monitored system load and predicted the interference using a mathematical regression model so as to aid in future task allocation. The model also predicted the optimal load in each server using the Pareto principle and threshold range. It also helped in scaling decisions for achieving better QoS. Thus the proposed model achieved automatic scaling that helped to handle sudden load changes with precise prediction and minimum VM migrations. Since there is only a rare chance of migrations, the system achieves stability that improved overall performance of the system compared to existing methods. We have tested the proposed method on the real cloud environment in five different workload conditions. We have tested and proved the accuracy of the prediction mechanism. The experimental results and comparative analysis validated the efficiency of the incorporated prediction mechanism in the cloud scheduling.

SLA enforcement through an auto scaling mechanism is proposed in chapter 7. It considered price variations, violations in deadline and response time as SLA parameter in the market oriented cloud. In this chapter, we have proposed scheduling and load balancing mechanism based on Petri Net model with auto scaling. Here we have utilized the properties of Petri Net to enhance the multi objective cloud scheduling mechanism. In addition, we have considered a dynamic spot pricing strategy with penalty if violations occur in the agreed conditions. The method also supports auto scaling to ensure QoS. To reduce frequent migrations, algorithm used principle of locality to reduce imbalance in the cloud. By simulation results and comparative analysis we have proved that the proposed model provides better performance in terms of time, cost and migrations.

Finally, the chapter 8 proposed an integrated quality assured SLA aware load balancing and scheduling algorithm for the cloud environment with cost consideration. This method considers processing power, memory requirement, bandwidth and cost as the SLA parameters. We have also proposed a prediction model based on the past usage pattern and that aims to provide optimal resource management without the violations of the agreed service level conditions in cloud datacenters. It considered SLA in both initial scheduling stage and in the load balancing stage and also, it looks into different objectives to achieve minimum makespan, minimum degree of imbalance and the minimum number of SLA violations with reduced cost. The experimental results proved the effectiveness of the proposed system compared to other state-of-art algorithms in terms of cost, makespan, SLA violations and stability.

Thus this thesis designed and developed QoS guaranteeing scheduling methodologies to improve cloud performance while considering makespan, stability with better load balancing, scalability, cost, and energy efficiency with service level agreements.

9.5 Future Directions

As one tries to derive the further directions of future research form the results summarized in the present thesis, it turns out that the scheduling and resource allocation in cloud is a live problem on account of the diverse requirements of applications and user needs. The world is moving towards Internet of Things (IoT) implementations. That data disseminated from these IoT devices is huge and more cloud implementations are needed to handle the data and applications. This scenario leads to the Cloud of Things (CoT) situation. The algorithms and methods developed in this is thesis can be extended to handle this scenario.

REFERENCES

1. Peter M. Mell and Timothy Grance, “The NIST definition of cloud computing”, Technical Report, SP 800-145, National Institute of Standards & Technology, USA 2011.
2. Rajkumar Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility”, Future Generation computer systems, Vol. 25, No. 6, pp.599–616, 2009. Elsevier.
3. Rajkumar Buyya, Christian Vecchiola and S. Thamarai Selvi, “Mastering Cloud Computing: Foundations and Applications Programming”, Waltham, MA: 2013, Elsevier.
4. “Google Apps”, Available: [Online]. https://apps.google.com/ (accessed on 24/12/2018).
5. “Google App Engine”, Available: [Online]. https: //appengine.google.com/ (accessed on 24/12/2018).
6. “Microsoft Azure”, Available: [Online]. https: //azure.microsoft.com/ (accessed on 24/12/2018).
7. “Oracle Cloud”, Available: [Online]. https://cloud.oracle.com/ (accessed on 24/12/2018).
8. “force.com”, Available: [Online]. https://www.salesforce.com/ (accessed on 24/12/2018).
9. “GoGrid”, Available: [Online]. https://www.rackspace.com/ (accessed on 24/12/2018).
10. “Amazon Elastic Cloud EC2”, Available: [Online]. https://aws.amazon.com/ec2/ (accessed on 24/12/2018).
11. “Eucalyptus cloud”, Available: [Online]. https://www.eucalyptus.cloud/ (accessed on 24/12/2018).

12. “Flexiscale cloud”, Available: [Online]. http://www.flexiscale.com/ (accessed on 24/12/2018).
13. “Rackspace cloud”, Available: [Online]. https://www.rackspace.com/ (accessed on 24/12/2018).
14. Linlin Wu and Rajkumar Buyya, "Service Level Agreement (SLA) in Utility Computing Systems", CoRR, pp.1-27, 2010. IGI.
15. Bhaskar Prasad Rimal and Martin Maier, “Workflow Scheduling in Multi-Tenant Cloud Computing Environments”, IEEE Transactions on Parallel and Distributed Systems, Vol.28, No.1, pp.290-304, 2017.
16. Walter Cerroni and Flavio Esposito, “Optimizing Live Migration of Multiple Virtual Machines”, IEEE Transactions on Cloud Computing, Vol. 6, Issue: 4, pp.1096 -1109, 2018.
17. Huandong Wang, Yong Li, Ying Zhang and Depeng Jin, “Virtual Machine Migration Planning in Software-Defined Networks”, Proceedings of the IEEE Conference on Computer Communication, pp.487-495, 2015. IEEE.
18. Yi Yao, Jiayin Wang, Bo Sheng, Chiu C. Tan and Ningfang Mi, “Self-Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters”, IEEE Transactions on Cloud Computing, Vol. 5, No. 2, pp.344-357, 2017.
19. Hamed Shah-Mansouri, Vincent W. S. Wong and Robert Schober, “Joint Optimal Pricing and Task Scheduling in Mobile Cloud Computing Systems”, IEEE Transactions on Wireless Communications, Vol. 16, No. 8, 2017.
20. Simon S. Woo and Jelena Mirkovic, “Optimal application allocation on multiple public clouds”, Computer Networks, Vol. 68, pp.138-148, 2014. Elsevier.
21. Haitao Yuan, Jing Bi, Wei Tan and Bo Hu Li, “Temporal Task Scheduling With Constrained Service Delay for Profit Maximization

in Hybrid Clouds", IEEE Transactions on Automation Science and Engineering, Vol. 14, No. 1, 2017.

22. Jincy Joseph and K.R. Remesh Babu, "Scheduling to Minimize Context Switches for Reduced Power Consumption and Delay in the Cloud", Proceedings of the International Conference on Micro-Electronics and Telecommunication Engineering, pp.545-549, 2016. IEEE.

23. Xianling Meng, Wei Wang and Zhaoyang Zhang, "Delay-Constrained Hybrid Computation Offloading with Cloud and Fog Computing", IEEE Access, Vol. 5, pp.21355-21367, 2017.

24. Hao Wu, Xiayu Hua, Zheng Li and Shangping Ren, "Resource and Instance Hour Minimization for Deadline Constrained DAG Applications Using Computer Clouds", IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 3, pp.885-899, 2016.

25. Kanchana Viriyapant and Sucha Smanchat, "A Deadline-constrained Scheduling for Dynamic Multi-instances Parameter Sweep Workflow", Proceedings of the 15th International Conference on Computer and Information Science (ICIS), pp.1-6, 2016. IEEE/ACIS.

26. Xiaoping Li, Lihua Qian and Rub´en Ruiz, "Cloud workflow scheduling with deadlines and time slot availability", IEEE Transactions on Services Computing, Vol.11, pp.329-340, Issue: 2, 2018.

27. Yaser Mansouri, Adel Nadjaran Toosi and Rajkumar Buyya, "Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers", IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2017.

28. Moussa Ehsan, Karthiek Chandrasekaran, Yao Chen and Radu Sion, "Cost-Efficient Tasks and Data Co-Scheduling with Afford Hadoop", IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2017.

29. Keke Gai, Meikang Qiu and Hui Zhao, "Cost-Aware Multimedia Data Allocation for Heterogeneous Memory Using Genetic

Algorithm in Cloud Computing", IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2016.

30. Sowmya Karunakaran and Rangaraja P. Sundarraj, "Bidding Strategies for Spot Instances in Cloud Computing Markets", IEEE Internet Computing, Vol. 19, Issue: 3, 2015.

31. Liang Zheng, Carlee Joe-Wong, Chee Wei Tan, Mung Chiang and Xinyu Wang, "How to Bid the Cloud", SIGCOMM'15, Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp.71-84, 2015.

32. Maristella Ribs, C.G.Furtado, José Neuman de Souza, Giovanni Cordeiro Barroso, Antão Moura, Alberto S Lima and Flávio R.C Sousa, "A Petri net-based decision-making framework for assessing cloud services adoption: The use of spot instances for cost reduction", Journal of Network and Computer Applications Vol. 57, pp.102-118, 2017. Elsevier.

33. PeiYun Zhang and Meng Chu Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy", IEEE Transactions on Automation Science and Engineering, Vol.15, Issue: 2, pp.772-783, 2018.

34. Lina Ni, Jinquan Zhang, Changjun Jiang, Chungang Yan and Kan Yu, "Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets", IEEE Internet of Things Journal, Vol. 4, No. 5, pp.772–783, 2017.

35. Yi-Li Zhang and Jin-Bai Zhang, "Schedule model in a cloud computing based on credit and cost", Computer Science, Technology and Application, pp.381-388, 2016. World Scientific.

36. Neethu B and K.R Remesh Babu, "Dynamic Resource Allocation in Market Oriented Cloud using Auction Method", Proceedings of International Conference on Micro-Electronics and Telecommunication Engineering, pp.145-150, 2016. IEEE.

37. Mohammad Aazam, Eui-Nam Huh, Marc St-Hilaire, Chung-Horng Lung and Ioannis Lambadaris, “Cloud Customer’s Historical Record Based Resource Pricing”, IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 7, pp.1929 -1940, 2016.

38. Songyun Wang, Zhuzhong Qian, Jiabin Yuan and Ilsun You, “A DVFS Based Energy-Efficient Tasks Scheduling in a Data Center”, IEEE Access, Vol. 5, pp.13090 -13102, 2017.

39. Yibin Li, Min Chen, Wenyun Dai and Meikang Qiu, “Energy Optimization With Dynamic Task Scheduling Mobile Cloud Computing”, IEEE Systems Journal, Vol. 11, No. 1, pp.96-105, 2017.

40. Hancong Duan, Chao Chen, Geyong Min and Yu Wu, “Energy-Aware Scheduling of Virtual Machines in Heterogeneous Cloud Computing Systems”, Future Generation Computer Systems, Vol. 74, pp.142-150, 2017. Elsevier.

41. Li Shi, Zhemin Zhang, and Thomas Robertazzi, “Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud”, IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 6, pp.1607-1620, 2017.

42. Weiwen Zhang and Yonggang Wen, “Energy-efficient Task Execution for Application as a General Topology in Mobile Cloud Computing”, IEEE Transactions on Cloud Computing,Vol.6, Issue: 3, pp.707-719, 2018.

43. Sonia Yassa, Rachid Chelouah, Hubert Kadima and Bertrand Granado, “Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments”, The Scientific World Journal, Vol. 2013, Article ID 350934, 13 pages, 2013. Hindawi.

44. R. K. Jena, “Multi Objective Task Scheduling in Cloud Environment Using Nested PSO Framework”, Procedia Computer Science, Vol. 57, pp.1219-1227, 2015, Elsevier.

45. Orachun Udomkasemsub, Li Xiaorong and Tiranee Achalakul, "A Multiple-Objective Workflow Scheduling Framework for Cloud Data Analytics", Proceedings of the Ninth International Joint Conference on Computer Science and Software Engineering (JCSSE), pp.391-398, 2012. IEEE.
46. Danlami Gabi, Abdul Samad Ismail, Anazida Zainal and Zalmiyah Zakaria, "Scalability-aware Scheduling Optimization Algorithm for Multi-Objective Cloud Task Scheduling Problem", Proceedings of the 6th ICT International Student Project Conference (ICT-ISPC), pp.1-6, 2017. IEEE.
47. K. Muralitharan R. Sakthivel and Y.Shi, "Multiobjective optimization technique for demand side management with load balancing approach in smart grid", Journal of Neuro computing, Vol. 177, pp.110-119, 2016. Elsevier.
48. Heyang Xu, Bo Yang, Weiwei Qi and Emmanuel Ahene, "A Multi-objective Optimization Approach to Workflow Scheduling in Clouds Considering Fault Recovery", KSII Transactions on Internet and Information Systems, Vol. 10, No. 3, pp.976-995, 2016.
49. C. Saravanakumar and C. Arun, "Efficient Idle Virtual Machine Management for Heterogeneous Cloud using Common Deployment Model", KSII Transactions on Internet and Information Systems Vol. 10, No. 4, pp.1501-1518, 2016.
50. Aissan Dalvandi, Mohan Gurusamy and Kee Chaing Chua, "Application Scheduling, Placement, and Routing for Power Efficiency in Cloud Data Centers", IEEE Transactions on Parallel and Distributed Systems, Vol. 28, Issue: 4, pp.947-960, 2017.
51. Shangguang Wang, Zhipiao Liu, Zibin Zheng, Qibo Sun and Fangchun Yang, "Particle Swarm Optimization for Energy-Aware Virtual Machine Placement Optimization in Virtualized Data Centers", Proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems, pp.102-109, 2013.

52. Konstantinos Tsakalozos, Vasilis Verroios, Mema Roussopoulos and Alex Delis, "Live VM Migration under Time-Constraints in Share-Nothing IaaS-Clouds", IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 8, pp.2285-2298, 2017.

53. Weiwei Kong, Yang Lei and Jing Ma, "Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism", International Journal Optik, Vol. 127, pp.5099-5104, 2016. Elsevier.

54. Zhifeng Zhong, Kun Chen, Xiaojun Zhai and Shuange Zhou, "Virtual Machine-Based Task Scheduling Algorithm in a Cloud Computing Environment", Tsinghua Science and Technology, Vol. 21, No. 6, pp.660-667, 2016. IEEE.

55. Seyed Ebrahim Dashti and Amir Masoud Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing", Journal of Experimental & Theoretical Artificial Intelligence, Vol. 28, Issue: 1-2, pp.97-112, 2016. Taylor & Francis.

56. Shaobin Zhan and Hongying Huo, "Improved PSO-based Task Scheduling Algorithm in Cloud Computing" Journal of Information & Computational Science, Vol. 9, No. 13, pp.3821-3829, 2012. World Academic Press.

57. Jianen Yan, Hongli Zhang, Haiyan Xu and Zhaoxin Zhang, "Discrete PSO-based workload optimization in virtual machine placement", Personal and Ubiquitous Computing, Vol. 22: 589, pp.589-596, 2018. Springer.

58. Yunliang Chen, Lizhe Wang, Xiaodao Chen, Rajiv Ranjan, Albert Y, Zomaya, Yuchen Zhou and Shiyan Hu, "Stochastic Workload Scheduling for Uncoordinated Datacenter Clouds with Multiple QoS Constraints", IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2016.

59. Nikos Tziritas, Samee U. Khan, Thanasis Loukopoulos, Spyros Lalis, Cheng-Zhong Xu, Keqin Li and Albert Y. Zomaya, "Online Inter-

Datacenter Service Migrations", IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2017.

60. Fahimeh Ramezani, Jie Lu and Farookh Khadeer Hussain, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization", International Journal of Parallel Programming, Vol. 42, Issue: 5, pp.739-754, 2014. Springer.

61. Hsu-Yang Kung, Ting-HuanKuo, Chi-Hua Chen and Yu-Lun Hsu, "Two-stage cloud service optimisation model for cloud service middleware platform", The Journal of Engineering, Vol. 2018, Issue: 3, pp. 155-161, 2018. IET.

62. Yadaiah Balagoni and Rajeswara Rao, "A Cost-effective SLA-Aware Scheduling for Hybrid Cloud Environment", Proceedings of the International Conference on Computational Intelligence and Computing Research, pp.15-17, 2016. IEEE.

63. Bahman Keshanchi and Nima Jafari Navimipour, "Priority-Based Task scheduling in the Cloud Systems Using a Memetic Algorithm", Journal of Circuits, Systems, and Computers, Vol. 25, No. 10, 2016. World Scientific.

64. P. K. Suri and Sunita Rani, "Simulator for Priority based Scheduling of Resources in Cloud Computing", International Journal of Computer Applications, Vol. 146, No.14, pp.10-15, 2016.

65. D. I. George Amalarethinam and S Kavitha, "Priority based Performance Improved Algorithm for Meta-task Scheduling in Cloud environment, Proceedings of the 2nd International Conference on Computing and Communications Technologies (ICCCT'17), pp.69-73, 2017. IEEE.

66. Mohamed Mohamed, Mourad Amziani, Djamel Belaïd, Samir Tata and Tarek Melliti, "An autonomic approach to manage elasticity of business processes in the Cloud", Future Generation Computer Systems , Vol. 50, pp.49-61, 2015. Elsevier.

67. Jiali You, Nannan Qiao, Jinlin Wang, Guoqiang Zhang,Yiqiang Sheng, Haojiang Deng and Xue Liu, “An On-Site Elastic Autonomous Service Network with Efficient Task Assignment”, Proceedings of the 41st Conference on Local Computer Networks Workshops, pp.42-29, 2016. IEEE.
68. Xiaomin Zhu, Ji Wang, Hui Guo, Dakai Zhu, Laurence T. Yang and Ling Liu, “Fault Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds”, IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 12, pp.3501-3517, 2016.
69. Kwang Mong Sim, “Agent-based Approaches for Intelligent Intercloud Resource Allocation”, IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2016.
70. Anwesha Mukherjee, Debashis De and Deepsubhra Guha Roy, “A Power and Latency Aware Cloudlet Selection Strategy for Multi-Cloudlet Environment”, IEEE Transactions on Cloud Computing, Vol. 7, Issue: 1, pp.141-154, 2019.
71. XQiu, Y Dai, Y Xiang and L Xing, “Correlation Modeling and Resource Optimization for Cloud Service with Fault Recovery”, IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), pp.1-13, 2017.
72. Xiaolin Chang, Ruofan Xia, Jogesh K. Muppala, Kishor S. Trivedi and Jiqiang Liu, “Effective Modeling Approach for IaaS Data Center Performance Analysis under Heterogeneous Workload”, IEEE Transactions on Cloud Computing, Vol. 6 Issue: 4, pp.991-1003, 2016.
73. Haiying Shen, “RIAL: Resource Intensity Aware Load Balancing in Clouds”, IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2017.
74. Binglai Niu, Yong Zhou, Hamed Shah-Mansouri and Vincent W. S. Wong, “A Dynamic Resource Sharing Mechanism for Cloud Radio

Access Networks", IEEE Transactions on Wireless Communications, Vol. 15, No. 12, pp. 8325-8338, 2016.

75. Cong Wang, Kui Ren and Jia Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing", Proceedings of INFOCOM' 11, pp.820-828, 2011. IEEE.

76. Ali Pahlevan, Xiaoyu Qu, Marina Zapater and David Atienza, "Integrating Heuristic and Machine-Learning Methods for Efficient Virtual Machine Allocation in Data Centers", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.37, Issue: 8, pp.1667-1680, 2018.

77. Cong Wang, Kui Ren and Jia Wang, "Secure Optimization Computation Outsourcing in Cloud Computing: A Case Study of Linear Programming", IEEE Transactions on Computers, Vol. 65, Issue: 1, pp.216-229, 2016.

78. Bin Xiang, Bibo Zhang and Lin Zhang, "Greedy-Ant: Ant Colony System-Inspired Workflow Scheduling for Heterogeneous Computing", IEEE Access, Vol. 5, pp.11404-11412, 2017.

79. Anton Beloglazov, Jemal Abawajy and Rajkumar Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Future Generation Computer Systems, Vol. 25, Issue: 5, pp.755-768, 2012. Elsevier.

80. Ali Al Buhussain, Robson E. De Grande and Azzedine Boukerche, "Elasticity Based Scheduling Heuristic Algorithm for Cloud Environments", Proceedings of the IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications, pp.1-8, 2016. IEEE.

81. Yacine Kessaci, Nouredine Melab and El-Ghazali Talbi, "A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager", Future Generation Computer Systems, Vol. 36, pp. 237-256, 2014. Elsevier.

82. Jia Zhao, Kun Yang, Xiaohui Wei, Yan Ding, Liang Hu and Gaochao Xu, “A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment”, IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 2, 2016.

83. Shengjun Xue, Wenling Shi, and Xiaolong Xu, “A Heuristic Scheduling Algorithm based on PSO in the Cloud Computing Environment”, International Journal of u- and e- Service, Science and Technology, Vol. 9, No. 1, pp.349-362, 2016. SERSC.

84. Syed Hamid Hussain Madni, Muhammad ShafieAbd Latiff, Mohammed Abdullahi, Shafi'i Muhammad Abdulhamid and Mohammed Joda Usman, “Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment”, PLOS ONE, Vol. 12, Issue: 5, pp.1-26, 2017.

85. Zhicheng Cai, Xiaoping Li and Jatinder N.D. Gupta, “Heuristics for Provisioning Services to Workflows in XaaS Clouds”, IEEE Transactions on Services Computing, Vol. 9, No. 2, 2016.

86. Chun-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang and Chu-Sing Yang, “A Hyper-Heuristic Scheduling Algorithm for Cloud”, IEEE Transactions on Cloud Computing, Vol. 2, No. 2, 2014.

87. Shengxiang Yang and Sadaf Naseem Jat, “Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling”, IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 41, No. 1, 2011.

88. Yonghua Xiong, Suzhen Huang, Min Wu, Jinhua She and Keyuan Jiang, “A Johnson’s-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing”, IEEE Transactions on Cloud Computing, Vol. PP, pp.(1-1), 2017.

89. Tawfeek MA, El-Sisi A, Keshk AE and Torkey FA, "Cloud task scheduling based on ant colony optimization", Proceedings of the 8th International Conference on Computer Engineering & Systems (ICCES), pp.64-69. 2013. IEEE.

90. Pacini E, Mateos C and Carlos García Garinoad, "Balancing throughput and response time in online scientific clouds via ant colony optimization", Advances in Engineering Software, Vol.84, Issue C, pp.31-47, 2015. Elsevier.

91. Li K, Xu G, Zhao G, Dong Y and Wang D, "Cloud task scheduling based on load balancing ant colony optimization", Proceedings of the Sixth Annual Chinagrid Conference, pp.3-9, 2011. IEEE.

92. Liu X, Zhan Z, Du K and Chen W, "Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization", Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO '14), pp.41-48, 2014. ACM.

93. Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N. Calheiros and Rajkumar Buyya, "Virtual machine consolidation in cloud data centers using ACO metaheuristic", Lecture Notes in Computer Science, Vol. 8632. Cham Proceedings of the Euro-Par 2014 parallel process, pp.306-317, 2014. Springer.

94. Y. Gao, H. Guan, Z. Qi, Y. Hou and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", Journal of Computer and System Sciences, Vol. 79, Issue: 8, pp.1230-1242, 2013. Elsevier.

95. Quanwang Wu, Fuyuki Ishikawa, Qingsheng Zhu, Yunni Xia and Junhao Wen, "Deadline-constrained Cost Optimization Approaches for Workflow Scheduling in Clouds", IEEE Transactions on Parallel and Distributed Systems, Vol. 28, Issue: 12, pp.3401-342, 2017.

96. Ashish Gupta and Ritu Garg, "Load Balancing Based Task Scheduling with ACO in Cloud Computing", Proceedings of the

International Conference on Computer an Applications (ICCA), pp.174-179, 2017. IEEE.

97. Asmae Benali, Bouchra El Asri and Houda Kriouile, “A Pareto-based Artificial Bee Colony and Product Line for Optimizing Scheduling of VM on Cloud Computing”, Proceedings of the International Conference on Cloud Technologies and Applications (CloudTech), pp.1-7, 2015. IEEE.

98. Kriti Agrawal and Priyanka Tripathi, “Power aware Artificial Bee Colony Virtual Machine Allocation for Private Cloud Systems”, Proceedings of the International Conference on Computational Intelligence and Communication Networks (CICN), pp.947-950, 2015. IEEE.

99. Warangkhana Kimpan and Boonhatai Kruekaew, “Heuristic Task Scheduling with Artificial Bee Colony Algorithm for Virtual Machines”, Proceedings of the Joint 8th International Conference on Soft Computing and Intelligent Systems and 17th International Symposium on Advanced Intelligent Systems, pp.281-286, 2016. IEEE.

100. An-ping Xiong and Chun-xiang Xu, “Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center,” Mathematical Problems in Engineering, Vol. 2014, pp.1-8, 2014. Hindawi.

101. Solmaz Abdi, Seyyed Ahmad Motamedi and Saeed Sharifian, “Task Scheduling using Modified PSO Algorithm in Cloud Computing Environment”, Proceedings of the International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014), pp.37-41, 2014. IEEE.

102. Entisar S. Alkayal, Nicholas R. Jennings and Maysoon F. Abulkhair, “Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing”, Proceedings of the 41st

Conference on Local Computer Networks Workshops, pp.17-24, 2016. IEEE.

103. Dinesh Kumar and Zahid Raza, “A PSO based VM Resource Scheduling Model for Cloud Computing”, Proceedings of the International Conference on Computational Intelligence & Communication Technology, pp.213-219, 2015. IEEE.

104. Liu Z and Wang X, “A PSO-based algorithm for load balancing in virtual machines of cloud computing environment”, Lecture Notes in Computer Science, ICSI 2012, Advances in Swarm Intelligence, Vol. 7331, pp.142-147, 2012. Springer.

105. Shahrzad Aslanzadeh and Zenon Chaczko, “Load balancing optimization in cloud computing: Applying Endocrine-particale swarm optimization”, Proceedings of the International Conference on Electro/Information Technology (EIT), pp.165-169, 2015. IEEE.

106. Juan J, Durillo, Vlad Nae and Radu Prodan, “Multi-objective energy-efficient workflow scheduling using list-based heuristics”, Future Generation Computer Systems, Vol. 36, pp. 221-236, 2014. Elsevier.

107. Jia Zhao, Liang Hu, Yan Ding, Gaochao Xu, Ming Hu, “A Heuristic Placement Selection of Live Virtual Machine Migration for Energy-Saving in Cloud Computing Environment”, PLoS ONE, 9(9), e108275, 2014.

108. Xingquan Zuo, Guoxiang Zhang and Wei Tan, “Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud”, IEEE Transactions on Automation Science and Engineering, Vol. 11, No. 2, pp.564-573, 2014.

109. Hua He, Guangquan Xu, Shanchen Pang and Zenghua Zhao, “AMTS: Adaptive multi-objective task scheduling strategy in cloud computing”, China Communications, Vol. 13, Issue: 4, pp. 162 -171, 2016. IEEE.

110. Maria Alejandra Rodriguez and Rajkumar Buyya, “Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific

Workflows on Clouds", IEEE Transactions on Cloud Computing, Vol. 2, Issue 2, pp. 222 -235, 2014.

111. Lizheng Guo, Shuguang Zhao, Shigen Shen and Changyuan Jiang, "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm, Journal of Networks", Vol. 7, No. 3, pp.547-553, 2012. Academy Publisher.

112. Pandey S, Wu L, Guru and Buyya R, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", Proceedings of the 24th International Conference on Advanced Information Networking and Applications, pp.400-407, 2010. IEEE.

113. Zhangjun Wu, Ni Z, Gu L and Liu X, "A revised discrete particle swarm optimization for cloud workflow scheduling", Proceedings of the International Conference on Computational Intelligence and Security, pp.184-188, 2010, IEEE.

114. Nazia Anwar and Huifang Deng, "A Hybrid Metaheuristic for Multi-Objective Scientific Workflow Scheduling in a Cloud Environment", Applied Sciences, 8, 538, 2018. MDPI.

115. Haitao Yuan, Jing Bi, Wei Tan, Meng Chu Zhou, Bo Hu Li and Jianqiang Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds", IEEE Transactions on Cybernetics, Vol. 47, No. 11, pp.3658-3668, 2017.

116. Gamal F. Elhady and Medhat A. Tawfeek, "A Comparative Study into Swarm Intelligence Algorithms for Dynamic Tasks Scheduling in Cloud Computing", Proceedings of the Seventh International Conference on Intelligent Computing and Information Systems (ICICIS'15), pp.362-269, 2015. IEEE.

117. Danlami Gabi and Abdul Samad Ismail, "Cloud Scalable Multi-Objective Task Scheduling Algorithm for Cloud Computing Using Cat Swarm Optimization and Simulated Annealing", Proceedings of

the 8th International Conference on Information Technology , pp.1007-1012, 2017. IEEE.

118. Keng-Mao Cho, Pang-Wei Tsai, Chun-Wei Tsai and Chu-Sing Yang, “A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing”, Neural Computing and Applications, Vol. 26, Issue: 6, pp.1297-1309, 2015. Springer-Verlag.
119. Wen X, Huang M and Shi J, “Study on resources scheduling based on ACO algorithm and PSO algorithm in cloud computing”, Proceedings of the 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, pp.219-222, 2012. IEEE.
120. Kanwarpreet Kaur and Amardeep Kaur, “A hybrid approach of load balancing through VMs using ACO, Min Max and genetic algorithm”, Proceedings of the 2nd International Conference on Next Generation Computing Technologies (NGCT), pp.615-620, 2016. IEEE.
121. Sheng-Jun Xue and Wu Wu, “Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm”, TELKOMNIKA, Vol.10, No.7, pp.1560-1566, 2012.
122. Shaobin Zhan and Hongying Huo, “Improved PSO-based Task Scheduling Algorithm in Cloud Computing”, Journal of Information & Computational Science, Vol. 9, No.13, pp.3821-3829, 2012.
123. D. Kusic, J. Kephart, J. Hanson, N. Kandasamy and G. Jiang, “Power and performance management of virtualized computing environments via lookahead control”, Proceedings of the International Conference on Autonomic Computing, pp.3-12, 2008. IEEE.
124. E. Bin, O. Biran, O. Boni, E. Hadad, E. Kolodner, Y. Moatti and D. Lorenz, “Guaranteeing high availability goals for virtual machine placement”, Proceedings of 31st International Conference on Distributed Computing Systems, pp.700 -709, 2011. IEEE.

125. J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing", Proceedings of the 9th International Conference on Grid and Cloud Computing (GCC 2010), pp.87-92, 2010. IEEE.
126. U. Sharma, P. Shenoy, S. Sahu and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud", Proceedings of 31st International Conference on Distributed Computing Systems, pp.559-570, 2011. IEEE.
127. S. Chaisiri, B.-S. Lee and D. Niyato, "Optimal virtual machine placement across multiple cloud providers", Proceedings of the IEEE Asia-Pacific Services Computing Conference, pp.103-110, 2009. IEEE.
128. Guofu Feng, Saurabh Garg, Rajkumar Buyya and Wenzhong Li, "Revenue maximization using adaptive resource provisioning in cloud computing environments", Proceeding GRID '12 Proceedings of the ACM/IEEE 13th International Conference on Grid Computing, pp. 192-200, 2012. ACM.
129. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Rajkumar Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, Vol. 41, Issue: 1, pp.23-50, 2011. John Wiley & Sons
130. Dervis Karaboga and Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Journal of Global Optimization, Vol. 39, pp.459-471, 2007. Springer.
131. Shridhar G. Domanal and G. Ram Mohana Reddy, "Load Balancing in Cloud Computing Using Modified Throttled Algorithm", Proceedings of International Conference on Cloud Computing in Emerging Markets (CCEM), pp.1-5, 2013. IEEE.

132. Shridhar G. Domanal and G. Ram Mohana Reddy, "Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines", Proceedings of Sixth International Conference on Communication Systems and Networks (COMSNETS), pp.1-4, 2014. IEEE.

133. Agraj Sharma and Sateesh K Peddoju, "Response Time Based Load Balancing in Cloud Computing", Proceedings of International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp.1287-1293, 2014. IEEE.

134. Gulshan Soni and Mala Kalra, "A Novel Approach for Load Balancing in Cloud Data Center", Proceedings of International Conference on Advance Computing Conference (IACC), pp.807-812, 2014. IEEE.

135. Santanu Dam, Gopa Mandal, Kousik Dasgupta and Paramartha Dutta, "An Ant Colony Based Load Balancing Strategy in Cloud Computing", Advanced Computing, Networking and Informatics-Vol. 2. Smart Innovation, Systems and Technologies, Vol. 28, pp.403-413, 2014. Springer.

136. Mohammadreza M., Amir M.R and Anthony T.C, "Cloud Light Weight: a New Solution for Load Balancing in Cloud Computing", Proceedings of International Conference on Data Science & Engineering (ICDSE), pp. 44-50, 2014. IEEE.

137. Gaochao Xu, Junjie Pang and Xiaodong Fu, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", Journal of Tsinghua Science and Technology, pp.34-39, 2013. IEEE.

138. M. Randles, D. Lamb and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing", WAINA '10 Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp.551-556, 2010. IEEE.

139. Jing Yao and Ju-hou He, "Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm", Proceedings of 8th International Conference on Computing Technology and Information Management (ICCM), pp.185-189, 2012. IEEE.
140. Pooja Samal and Pranati Mishra, "Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing", International Journal of Computer Science and Information Technologies, Vol. 4, No. 3, pp. 416-419, 2013.
141. Ajit. M. and Vidya. G., "VM Level Load Balancing in Cloud Environment", Proceedings of IEEE Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1-5, 2013. IEEE.
142. Youssef Fahim, Elhabib Ben Lahmar, El houssine Labriji and Ahmed Eddaoui, "The load balancing based on the estimated finish time of tasks in cloud computing", Proceedings of 2nd World Conference on Complex Systems (WCCS), pp.594-598, 2014. IEEE.
143. Rakesh Madivi and S Sowmya Kamath, "An hybrid bio-inspired task scheduling algorithm in cloud environment", Proceedings of International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-7, 2014. IEEE.
144. Remesh Babu K.R., P Mathiyalagan and S.N. Sivanandam, "Pareto based hybrid Meta heuristic ABC – ACO approach for task scheduling in computational grids", International Journal of Hybrid Intelligent Systems, Vol. 11, No. 4, pp.241-255, 2014. IOS Press.
145. Ling Wang, Gang Zhou, Y Xu and M Liu, "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling", International Journal of Advanced Manufacturing Technology, Vol. 60, Issue: 9-12, pp. 1111-1123, 2012. Springer.
146. Artificial Bee Colony Algorithm. Home page. http://mf.erciyes.edu.tr/abc/.:, Retrieved on 15th January 2016.

147. Dhinesh Babu L.D and P. Venkata Krishna, “Bee behavior inspired load balancing of tasks in cloud computing environments”, Applied Soft Computing, Vol. 13, Issue: 5, pp.2292-2303, 2013. Elsevier.

148. Amir Nahir, Ariel Orda and Danny Raz, “Replication-Based Load Balancing”, IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No.2, pp.494-507, 2016.

149. Feilong Tang, Laurence T. Yang, Can Tang, Jie Li and Minyi Guo, “A Dynamical and Load-Balanced Flow Scheduling Approach for Big Data Centers in Clouds”, IEEE Transactions on Cloud Computing, Vol.6, Issue: 4, pp.915-928, 2018.

150. J. Octavio Gutierrez-Garcia and Adrian Ramirez-Nafarrate, "Collaborative Agents for Distributed Load Management in Cloud Data Centers Using Live Migration of Virtual Machines", IEEE Transactions on Services Computing, Vol.6, No.8, pp.916-929, 2015.

151. Shang-Liang Chen, Yun-Yao Chen and Suang-Hong Kuo, “CLB: A novel load balancing architecture and algorithm for cloud services”, Computers & Electrical Engineering, Vol. 58, pp.154-160, 2017. Elsevier.

152. Haitao Yuan, Jing Bi, Wei Tan and Bo Hu Li, “Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds”, IEEE Transactions on Automation Science and Engineering, Vol. 14, Issue: 1, pp.337-348, 2017.

153. W. K. Hsieh, W. H. Hsieh, J. L. Chen and P. J. Yang, “CssQoS: A load balancing mechanism for cloud serving system”, Proceedings of Information Technology and Applications, pp.269-275, 2015. CRC Press.

154. Rui Zhang, Kui Wu, Minming Li and Jianping Wang, “Online Resource Scheduling Under Concave Pricing for Cloud Computing”, IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No.4, pp. 1131-1145, 2016.

155. P. R. Nuth and W. J. Dally, "A mechanism for efficient context switching", Proceedings of International Conference on Computer Design: VLSI in Computers and Processors (ICCD '91), pp. 301-304, 1991. IEEE.
156. Yamini R, "Power management in cloud computing using green algorithm", Proceedings of International Conference on Advances in Engineering, Science and Management (ICAESM), pp.128-133, 2012. IEEE.
157. Bei Guan, Jingzheng Wu, Yongji Wang and Samee U. Khan, "Civsched: A communication-aware inter-VM scheduling technique for decreased network latency between collocated VMs", IEEE Transactions on Cloud Computing, Vol. 2, Issue: 3, pp.320-332, 2014.
158. A. Bharathi, R. S. Mohana and A. Ushapriya, "Profit and energy aware scheduling in cloud computing using task consolidation", Proceedings of International Conference on Information Communication and Embedded Systems (ICICES), pp.1-6, 2014. IEEE.
159. Sampa Sahoo, Bibhudatta Sahoo and Ashok Kumar Turuk, "An Energy-efficient Scheduling Framework for Cloud Using Learning Automata", Proceedings of 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp.1-5, 2018. IEEE.
160. B. Zolfaghari, "A dynamic scheduling algorithm with minimum context switches for spacecraft avionics systems", Proceedings of Aerospace Conference, Vol.4, pp.2618-2624, 2004. IEEE.
161. Alahmadi, A., D. Che, M. Khaleel, M. M. Zhu and P. Ghodous, "An innovative energy-aware cloud task scheduling framework", Proceedings of 8th International Conference on Cloud Computing (CLOUD), pp.493-500, 2015. IEEE.

162. G. Kaur, R. K. Bedi and S. K. Gupta, “Design and implementation of enhanced MQS algorithm”, Proceedings of International Conference on Green Computing and Internet of Things (ICGCIoT), pp.470-473, 2015. IEEE.

163. A. V. Karthick, E. Ramaraj and R. G., Subramanian, “An Efficient Multi Queue Job Scheduling for Cloud Computing”, Proceeding of World Congress on Computing and Communication Technologies (WCCCT), pp.164-166, 2014. IEEE.

164. Ajoy K. Datta and Rajesh Patel, “CPU Scheduling for Power/Energy Management on Multicore Processors Using Cache Miss and Context Switch Data”, IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 5, pp. 1190-1199, 2014.

165. A. V. Karthick, E. Ramaraj and R. Kannan, “An efficient Tri Queue job Scheduling using dynamic quantum time for cloud environment”, Proceedings of International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), pp. 871-876, 2013, IEEE.

166. Alnowiser, E. Aldhahri, A. Alahmadi and M. M. Zhu, “Enhanced Weighted Round Robin (EWRR) with DVFS Technology in Cloud Energy-Aware”, Proceedings of International Conference on Computational Science and Computational Intelligence (CSCI), Vol. 1, pp. 320-326, 2014. IEEE.

167. M. Dayarathna, Y. Wen and R. Fan, “Data Center Energy Consumption Modeling: A Survey”, IEEE Communications Surveys & Tutorials, Vol. 18, No. 1, pp.732-794, 2016.

168. Lee, Y.C. and Albert Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems”, The Journal of Supercomputing, Vol. 60, Issue: 2, pp.268-280, 2012. Springer.

169. Haihua Chang and Xinhuai Tang, “A Load-Balance Based Resource-Scheduling Algorithm under Cloud Computing Environment”, Proceedings New Horizons in Web-Based Learning - ICWL 2010

Workshops, Lecture Notes in Computer Science, Vol. 6537, pp.85-90, 2010. Springer.

170. Hong He, “Virtual resource provision based on elastic reservation in cloud computing”, International Journal of Networking and Virtual Organisations, Vol. 15, No. 1, pp.30-47, 2015. Inderscience.

171. Fahimeh Ramezani, Jie Lu and Farookh Khadeer Hussain, “Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization”, International Journal of Parallel Programming, Vol. 42, Issue: 5, pp.739-754, 2014. Springer.

172. Arpan Roy, Rajeshwari Ganesan and Santonu Sarkar, “Keep It Moving: Proactive workload management for reducing SLA violations in large scale SaaS clouds”, Proceedings of 24th International Symposium on Software Reliability Engineering (ISSRE), pp.421-430, 2013. IEEE.

173. Jian Guo, Fangming Liu, Haowen Tang, Yingnan Lian, Hai Jin and John C.S. Lui, “Falloc: Fair network bandwidth allocate on in IaaS datacenters via a bargaining game approach”, Proceedings of 21st International Conference on Network Protocols (ICNP), pp.7-10, 2013. IEEE.

174. Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li and Baochun Li, “iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud”, IEEE Transactions on Computers, Vol. 63, Issue: 12, pp.3012-3025, 2014.

175. Christopher D Wickens, “Multiple resources and performance prediction”, Theoretical Issues in Ergonomics Science, Vol.3, No.2, pp.159-177, 2002. Taylor & Francis.

176. Samer Al-Kiswany, Dinesh S, P Sarkar and M Ripeanu, “VMFlock: Virtual Machine Co-Migration for the Cloud”, Proceedings of the 20th International Symposium on High performance Distributed Computing, HPDC '11, pp.159-170, 2011. IEEE.

177. Frank Yong-Kyung Oh, Hyeong S. Kim, Hyeonsang Eom and Heon Y. Yeom, “Enabling consolidation and scaling down to provide power management for cloud computing”, Proceedings of the 3rd USENIX conference on Hot topics in cloud computing, HotCloud’11, pp.14-14, 2011. USENIX.

178. Yefu Wang and Xiaorui Wang, “Performance-controlled server consolidation for virtualized data centers with multi-tier applications”, Sustainable Computing: Informatics and Systems, Vol. 4, Issue: 1, pp. 52-65, 2014. Elsevier.

179. Qian Zhu, Jiedan Zhu and Gagan Agrawal, “Power-aware consolidation of scientific workflows in virtualized environments”, SC'10: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp.1-12, 2010. IEEE.

180. Constantine P. Sapuntzakis, Ramesh Chandra, Ben Pfaff, Jim Chow, Monica S. Lam, and Mendel Rosenblum, “Optimizing the migration of virtual computers”, Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02), ACM SIGOPS Operating Systems Review, Vol.36, Issue: SI, pp.377-390, 2002. ACM.

181. Seung-Hwan Lim, Jae-Seok Huh, Youngjae Kim and Chita R. Das, “Migration, assignment, and scheduling of jobs in virtualized environment”, Proceedings of the 3rd USENIX conference on Hot topics in cloud computing, HotCloud'11, pp.2-2, 2011. USENIX.

182. A Koto, H Yamada, K Ohmura and K Kono, “Towards unobtrusive VM live migration for cloud computing platforms”, Proceedings of the Asia-Pacific Workshop on Systems (APSys’12), Article No.7, pp.1-6, 2012. ACM.

183. Tao Lu, Ping Huang, Morgan Stuart, Yuhua Guo, Xubin He and Ming Zhang, “Successor: Proactive cache warm-up of destination hosts in virtual machine migration contexts”, Proceedings of the 35th

Annual International Conference on Computer Communications Computer Communications, INFOCOM 2016, pp.1-9, 2016. IEEE.

184. Woonghee Tim Huh, Nan Liu and Van-Anh Truong, "Multiresource Allocation Scheduling in Dynamic Environments", Manufacturing & Service Operations Management, Vol. 15, No. 2, pp.280-291, 2013. INFORMS.

185. V H Nguyen, S Khaddaj, A Hoppe and Eric Oppong, "A QoS Based Load Balancing Framework for Large Scale Elastic Distributed Systems", Proceedings of 10th IEEE International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), pp.146-150, 2011. IEEE.

186. Jun Chen, Yunchuan Qin, Yu Ye and Zhuo Tang, "A Live Migration Algorithm for Virtual Machine in a Cloud Computing Environment", Proceedings of IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and IEEE 12th Intl Conf on Autonomic and Trusted Computing and IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), pp.1319-1326, 2015.

187. Felipe Fernandes, David Beserra, Edward David Moreno, Bruno Schulze and Raquel Coelho Gomes Pinto, "A VM Scheduler Based in CPU and IO-bound Features for Energy-aware in High Performance Computing Clouds", Computers & Electrical Engineering, Vol. 56, pp.854-870, 2016. Elsevier.

188. A Jakóbik, D Grzonka and J Kołodziej, "Security Supportive Energy Aware Scheduling and Scaling for Cloud Environments", Proceedings of 31st European Conference on Modelling and Simulation, pp.583-590, 2017. ECMS.

189. Dang Tran, Nhuan Tran, Giang Nguyen and Binh Minh Nguyen, "A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness", Procedia Computer Science, Vol. 108, pp.365-374, 2017. Elsevier.

190. Michael T Krieger, Oscar Torreno, Oswaldo Trelles and Dieter Kranzlmüller, “Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows”, Future Generation Computer Systems , Vol. 67, pp. 329-340, 2017. Elsevier.
191. R C Chiang and H H Huang, “Tracon: interference-aware scheduling for data intensive applications in virtualized environments”, IEEE Transactions on Parallel and Distributed Systems, Vol. 25, Issue: 5, pp.1349-1358, 2014.
192. Auto Scaling: User guide, http://docs.aws.amazon.com /autoscaling/latest/userguide/as-dg.pdf, retrieved on 6th March 2017.
193. Akshat Verma, Puneet Ahuja and Anindya Neogi, “pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems”, Proceedings of 9th ACM/IFIP/USENIX International Conference on Middleware, pp.243-264, 2008. Springer.
194. Xiaofang Li, Yingchi Mao, Xianjian Xiao and Yanbin Zhuang, “An Improved Max-Min Task-Scheduling Algorithm for Elastic Cloud”, Proceedings of International Symposium on Computer, Consumer and Control (IS3C), pp.340-343, 2014. IEEE.
195. Huankai Chen, Frank Wang, Na Helian and Gbola Akanmu, “User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing”, Proceedings of 2013 IEEE National Conference on Parallel Computing Technologies (PARCOMPTECH), pp. 1-8, 2013. IEEE.
196. Tadao Murata, “Petri Nets: Properties, Analysis and Applications”, Proceedings of the IEEE, Vol. 77, No.4, pp.541-580. IEEE.
197. Amazon pricing history, Available [Online]. http://web.archive.org/web/20130601050642/http://aws.amazon.com: 80/ec2/pricing/
198. M Mohamed, M Amziani, D Belaid and S Tata, “An Autonomic Approach to Manage Elasticity of Business Processes in the Cloud”,

Future Generation Computer Systems, Vol.50, pp.49-61, 2015. Elsevier.

199. W Kong, Y Lei and Jing Ma, "Virtual Machine Resource Scheduling Algorithm for Cloud Computing based on Auction Mechanism", Journal of Optik, Vol. 127, Issue: 12, pp.5099-5104, 2016. Elsevier.

200. Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru and Rajkumar Buyya, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", Proceedings of 24th International Conference on Advanced Information Networking and Applications, pp.400-407, 2010. IEEE.

201. Amit Nathani, Sanjay Chaudhary and Gaurav Somani, "Policy based resource allocation in IaaS cloud", Future Generation Computer Systems", Vol. 28, Issue: 1, pp.94-103, 2012. Elsevier.

202. Bobroff N, Kochut A and Beaty K A, "Dynamic placement of virtual machines for managing SLA violations", Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, pp.119-128, 2007. IEEE.

203. Bowen Zhou, Satish Narayana Srirama and Rajkumar Buyya, "An Auction-based Incentive Mechanism for Heterogeneous Mobile Clouds", Journal of Systems and Software, Vol. 152, pp.151-164, 2019. Elsevier.

204. "Gugoos Cloud", Available: [Online]. http://www.gungoos.com/ (Accessed on 06/01/2017).

205. Anita Choudhary, M. C. Govil, Girdhari Singh and Lalit K. Awasthi, "Energy-Efficient Resource Allocation Approaches with Optimum Virtual Machine Migrations in Cloud Environment", Proceedings of 4th International Conference on Parallel, Distributed and Grid Computing (PDGC), pp.182-187, 2016. IEEE.

206. W. M. P. Van Der Aalst, "The Application of Petri Nets to Workflow Management", Journal of Circuits, Systems and Computers, Vol. 08, No.1, pp.21-66, 1998. World Scientific.
207. Yiming Han and Anthony T. Chronopoulos, "Scalable Loop Self-Scheduling Schemes for Large-Scale Clusters and Cloud Systems", International Journal of Parallel Programming, Vol. 45, Issue: 3, pp.595-611, 2017. Springer.
208. Cloud exchange website, Available [online]. https://cloudxchange.io/
209. Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan and Yun Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems", Journal of Supercomputing, Vol. 63, Issue: 1, pp. 256-293, 2013. Springer.
210. Salehi M A and Rajkumar Buyya, "Adapting market-oriented scheduling policies for cloud computing", Proceedings of the 10th International conference on Algorithms and Architectures for Parallel Processing, LNCS, Vol. Part I, pp.351-362, 2010. Springer.
211. Xuelin Shi, Ke Xu, Jiangchuan Liu and Yong Wang, "Continuous Double Auction Mechanism and Bidding Strategies in Cloud Computing Markets", CoRR abs/1307.6066 , 2013. ArXiv.
212. Shifeng Shang, Jinlei Jiang, Yongwei Wu, Guangwen Yang and Weimin Zheng, "A Knowledge-based Continuous Double Auction Model for Cloud Market", Proceedings of 6th International Conference on Semantics Knowledge and Grid, pp.129-134, 2010. IEEE.
213. M. Macías and J. Guitart, "Using Resource-level Information into Non additive Negotiation Models for Cloud Market Environments", Proceedings of Network Operations and Management Symposium, pp.325-332, 2010. IEEE.
214. P. Samimia, Y. Teimourib and M. Mukhtara, "A Combinatorial Double Auction Resource Allocation Model in Cloud Computing", Information Sciences, Vol. 357, No.20, pp.201-216, 2016. Elsevier.

215. Patricia Arroba, José M. Moya, José L. Ayala and Rajkumar Buyya, "Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers", Concurrency and Computation: Practice & Experience, Vol. 29, Issue: 10, e4067, 2017. John Wiley & Sons.

216. Georgios L. Stavrinides and Helen D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations", Future Generation Computer Systems, Vol. 96, pp.216-226, 2019. Elsevier.

217. Adel Nadjaran Toosi, Chenhao Qua, Marcos Dias de Assunção and Rajkumar Buyyaa, "Renewable-aware Geographical Load Balancing of Web Applications for Sustainable Data Centers", Journal of Network and Computer Applications, Vol.83, pp.155-168, 2017. Elsevier.

218. Yousseffahim, Elhabib Ben Lahmar,El Houssine Labrlji and Ahmed Eddaoui, "The load balancing based on the estimated finish time of tasks in cloud computing", Proceedings of Second World Conference on Complex Systems (WCCS 2014), pp. 594-598, 2014. IEEE.

219. Yu Liu, Changjie Zhang, Bo Li and Jianwei Niu, "DeMS : A hybrid scheme of task scheduling and load balancing in computing clusters", Journal of Network and Computer Applications, Vol. 83, pp.213-220, 2017. Elsevier.

220. Chenhao Qu, Rodrigo Neves Calheiros and Rajkumar Buyya, "Mitigating impact of short-term overload on multi-cloud web applications through geographical load balancing", Concurrency and Computation: Practice & Experience, Vol. 29, No.12, e4126, 2017. John Wiley & Sons.

221. Shaymaa Elsherbiny, Eman Eldaydamony, Mohammed Alrahmawy and Alaa Eldin Reyad, "An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment",

Egyptian Informatics Journal, Vol.19, Issue: 1, pp.33-55, 2018. Elsevier.

222. Amir Vahid Dastjerdi1, Sayed Gholam Hassan Tabatabaei and Rajkumar Buyya, “A dependency-aware ontology-based approach for deploying service level agreement monitoring services in Cloud”, Software Practice & Experience, Vol.42, Issue:4, pp.501-518, 2012. John Wiley & Sons.
223. Uri Lublin and Dror G. Feitelson, “The workload on parallel supercomputers: modeling the characteristics of rigid jobs”, Journal of Parallel and Distributed Computing”, Vol. 63, Issue: 11, pp.1105-1122. Elsevier.
224. Fan Zhang, Junwei Cao, Kai Hwang, Keqin Li, and Samee U. Khan, “Adaptive Workflow Scheduling on Cloud Computing Platforms with Iterative Ordinal Optimization”, IEEE Transactions on Cloud Computing, Vol. 3, No.2, pp.156-168, 2015.
225. Jinzhao Liu, Yaoxue Zhang, Yuezhi Zhou, Di Zhang, and Hao Liu, “Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments”, IEEE Transactions on Cloud Computing, Vol.3, No.2, pp.119-131, 2015.
226. Alok Gautam Kumbhare, Yogesh Simmhan, Marc Frincu and Viktor K. Prasanna, “Reactive Resource Provisioning Heuristics for Dynamic Dataflows on Cloud Infrastructure”, IEEE Transactions on Cloud Computing, Vol.3, No.2, pp. 105-118, 2015.
227. Maria Alejandra Rodriguez and Rajkumar Buyya, “A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments”, Concurrency and Computation: Practice & Experience, Vol.29, Issue: 8, e4044, 2016. John Wiley & Sons.
228. Abdelzahir Abdelmaboud, Dayang N.A. Jawawi, Imran Ghani, Abubakar Elsafi and Barbara Kitchenham, “Quality of service approaches in cloud computing: A systematic mapping study”, The

Journal of Systems and Software, Vol. 101, pp.159-179, 2015. Elsevier.

229. Tom Guérout, Samir Medjiah, Georges Da Costa and Thierry Monteil, “Quality of service modeling for green scheduling in Clouds”, Sustainable Computing: Informatics and Systems, Vol.4, Issue: 4, pp.225-240, 2014. Elsevier.

230. Sukhpal Singh and Inderveer Chana, “Q-aware: Quality of service based cloud resource provisioning”, Computers and Electrical Engineering, Vol. 47, pp.138-160, 2015. Elsevier.

231. David Breitgand and Amir Epstein, “SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds”, Proceedings of 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp.161-168, 2011. IEEE.

232. Leone FC, Nelson LS and Nottingham RB, “The folded normal distribution”, Technometrics, Vol. 3, No.4, pp.543-550, 1961. Taylor & Francis.

233. Patel, P., Ranabahu, A. H and Sheth A. P, “Service Level Agreement in Cloud Computing”, 2009. Source: http://corescholar.libraries.wright.edu/knoesis/78. Kno.e.sis Publications.

234. Deborah Magalhaes, Rodrigo N. Calheiros, Rajkumar Buyya and Danielo G. Gomes, “Workload modeling for resource usage analysis and simulation in cloud computing”, Computers and Electrical Engineering, Vol. 47, pp.69-81, 2015. Elsevier.

235. Ashkan Paya and Dan C Marinescu, “Energy-aware Load Balancing and Application Scaling for the Cloud Ecosystem”, IEEE Transactions on Cloud Computing, Vol. 5, Issue: 1, pp.15-27, 2017.

236. Guisheng Fan, Liqiong Chen, Huiqun Yu and Dongmei Liu, “Formally modeling and analyzing cost-aware job scheduling for cloud data center”, Software Practice & Experience, Vol. 49, Issue: 9, pp.1536-1559, 2018. John Wiley & Sons.

237. Yiqiu Fang, Fei Wang and Junwei Ge, “A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing”, International Conference on Web Information Systems and Mining. WISM 2010. Lecture Notes in Computer Science, Vol. 6318, pp.271-277, 2010. Springer.
238. Jinhua Hu, Jianhua Gu, Guofei Sun and Tianhai Zhao, “A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment”, Proceedings of 3rd International Symposium on Parallel Architectures, Algorithms and Programming, pp.89-96, 2010. IEEE.

LIST OF PUBLICATIONS BASED ON THIS THESIS

International Journals

1. K R Remesh Babu, Philip Samuel**, “Service-level agreement–aware scheduling and load balancing of tasks in cloud”**, Software: Practice and Experience, Vol. 49, No. 6, pp.995-1012, 2019. John Wiley & Sons.

2. K R Remesh Babu, Philip Samuel, **“Interference aware prediction mechanism for auto scaling in cloud”**, Computers and Electrical Engineering, Vol. 69, pp.351-363, 2018. Elsevier.

3. K R Remesh Babu, Philip Samuel, **“Energy aware clustered load balancing in cloud computing environment”**, International Journal of Networking and Virtual Organisations, Vol.19, No.2/3/4, pp.305-320, 2018. Inderscience.
4. K R Remesh Babu, Philip Samuel, **“Review of the quality of service scheduling mechanisms in cloud”**, International Journal of Engineering & Technology, Vol. 7, No. 3, pp.1677-1695, 2018. SPC.
5. K R Remesh Babu, Philip Samuel, **“Enhanced Resource Scheduling with Autoscaling in Elastic Cloud”**, International Journal of Networking and Virtual Organisations. (Accepted). 2019. Inderscience.

International Conferences

6. K R Remesh Babu, Philip Samuel, **"Virtual Machine Placement for Improved Quality in IaaS Cloud"**, In proceedings of the 4th International Conference on Advances in Computing and Communications, pp.190-194, 2014. IEEE.
7. K R Remesh Babu, Philip Samuel, **"Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud"**, Advances in Intelligent Systems and Computing, Vol. 424, pp. 67-78, 2015. Springer Verlag.