

**SMART: SELF-MANAGEMENT AWARE AUTONOMIC
RESOURCE MANAGEMENT TECHNIQUE IN CLOUD**

A thesis submitted to the
University of Petroleum and Energy Studies

For the Award of

Doctor of Philosophy

in

Computer Science and Engineering

by

Bhupesh Kumar Dewangan

JULY 2020

Supervisor(s)

Dr. Amit Agarwal

Dr. Tanupriya Choudhury

Dr. Ashutosh Pasricha



UNIVERSITY WITH A PURPOSE

School of Computer Science

University of Petroleum and Energy Studies

Dehradun, Uttarakhand-248007, India

**SMART: SELF-MANAGEMENT AWARE AUTONOMIC
RESOURCE MANAGEMENT TECHNIQUE IN CLOUD**

A thesis submitted to the
University of Petroleum and Energy Studies

For the Award of

Doctor of Philosophy

in

Computer Science and Engineering

by

Bhupesh Kumar Dewangan

(SAP ID 500042351)

JULY 2020

Internal Supervisor

Dr. Amit Agarwal

Professor (On Leave), SoCS, UPES, Dehradun

Internal Co-Supervisor

Dr. Tanupriya Choudhury

Associate Professor, SoCS, UPES, Dehradun

External Supervisor

Dr. Ashutosh Pasricha

OFS Director, Schlumberger Asia Services Ltd., India



UNIVERSITY WITH A PURPOSE

School of Computer Science

University of Petroleum and Energy Studies

Dehradun, Uttarakhand-248007, India

I dedicate my Ph.D. Thesis to

My loving Parents, In-Laws, and my Supervisors

Dr. Amit Agarwal,

Dr. Tanupriya Choudhury and

Dr. Ashutosh Pasricha

for their endless support, blessings and guidance.

DECLARATION

I declare that this thesis, which I submit to University of Petroleum and Energy Studies, Dehradun, for examination in consideration of the award of a higher degree Doctor of Philosophy in Computer Science and Engineering is my own personal effort. Where any of the content presented is the result of input or data from a related collaborative research programme this is duly acknowledged in the text such that it is possible to ascertain how much of the work is my own. Furthermore, I took reasonable care to ensure that the work is original, and, to the best of my knowledge, does not breach copyright law, and has not been taken from other sources except where such work has been cited and acknowledged within the text.



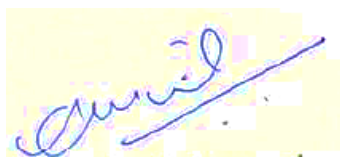
Signature of the Candidate

SAP ID: 500042351

Date:09 / 07 / 2020

THESIS COMPLETION CERTIFICATE

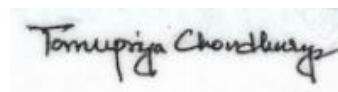
This is to certify that the thesis entitled "**SMART: Self-Management Aware Autonomic Resource Management Technique in Cloud**" by **Bhupesh Kumar Dewangan, (SAP ID: 500042351)** in partial completion of the requirements for the award of the Degree of Doctor of Philosophy in Computer Science and Engineering is an original work carried out by him under our joint supervision and guidance. It is certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.



Dr. Amit Agarwal

Internal Supervisor

Professor, SoCS, UPES.



Dr. Tanupriya Choudhury

Internal Co-Supervisor

Associate Professor, SoCS, UPES.

THESIS COMPLETION CERTIFICATE

This is to certify that the thesis entitled "**SMART: Self-Management Aware Autonomic Resource Management Technique in Cloud**" by **Bhupesh Kumar Dewangan, (SAP ID: 500042351)** in partial completion of the requirements for the award of the Degree of Doctor of Philosophy in Computer Science and Engineering is an original work carried out by him under our joint supervision and guidance. It is certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.



Dr. Ashutosh Pasricha

External Supervisor

Schlumberger Asia Service Ltd., Gurgaon, India

Abstract

Cloud computing plays the biggest role in modern Information Technology for the growth of the business. It is responsible to make the availability of numerous services like hardware, software, network, servers, etc. Resource management technique is one of the important methods to ensure to schedule these services to cloud users. For efficient resource management, and optimal resource to compute the required task is very important. Resource optimization is directly related to service level agreement SLA and associated hardware-software cost in the cloud. The effectiveness of any load-balancing algorithm is directly related to the utilization of infrastructure. The efficient utilization of resources will turn in satisfaction of service level agreement as well as profit for the service providers. In the cloud environment, the satisfaction of SLA is a prime objective. It can be achieved by providing services in a minimum time in an efficient manner at the lowest cost by efficiently utilizing the resources. This will create a win-win situation for both consumers and service providers.

In this thesis, a novel resource management technique self-management aware autonomic resource management technique in cloud SMART is proposed, which focus to optimize and maximize resource utilization, while maintaining lower execution time and cost. The objective of this work is to design a resource management system to improve the fault tolerance mechanism, self-adaptability, and maximize resource utilization in cloud computing. To achieve this, a new resource management algorithm named SMART is proposed by picking the best features from Antlion optimizers. Based on the availability of resources and workloads on a virtual machine, a fitness value is assigned to all virtual machines. A newly arrived task is mapped with the fittest virtual machine. Whenever a new task is mapped or left the system, the fitness value of the virtual machine is updated. In this manner, the system achieves the satisfaction of service level agreement, the balance of workload, and efficient utilization of resources. To test the proposed approach, the

Cloudsim 3.0 simulator has been used, and then it is validated in the real-time cloud environment (Amazon web services) as well. Through experimental results, it can be concluded that the proposed resource management approach SMART has outperformed other resource management approach based on relevant parameters.

Acknowledgement

I bow my head humbly to pay heart felt regards to Almighty God for giving me the strengths and blessing in completing this thesis.

There are quite a few people that have helped me in one way or another to the completion of this work. It is with great pleasure, I would like to thank all of you from very deep inside.

Foremost, I would like to express my sincere gratitude to my thesis advisors Dr. Amit Agarwal, Dr. Tanupriya Choudhury and Dr. Ashutosh Pasricha, for picking me up as a student at the critical stage of my career and the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

It is absolutely difficult to succeed in the process of finding and developing an idea without the help of a specialist in the domain. I found in my advisors not only the source of wonderful ideas to develop, but also the support that a Ph.D student needs. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisors, I would like to thank Chancellor Dr. S. J. Chopra, Vice chancellor Dr. Sunil Rai and Dean SoCS Dr. Manish Prateek, Dean SoCS R&D Dr. Kiran Kumar Ravulakollu at the University of Petroleum and Energy Studies for their encouragement, suggestions and valuable support for my research work.

I would like to express my special thanks to Dr. J K Pandey, R&D, Director and Dr. Rakhi Ruhel, Program Manager-Ph.D, University of Petroleum and Energy Studies, for his assistance during my research work. I am grateful to the University

of Petroleum and Energy Studies, for giving me an opportunity to pursue my research and for providing all facilities in School of Computer Science.

I would like to thank all the Heads of School of Computer Science Departments, doctoral students for their feedback, cooperation, and of course friendship. In addition I would like to express my gratitude to all colleagues in the university.

I cannot begin to express my gratitude to my family for all of the love, support, encouragement, and prayers they have sent my way along this journey. I am eternally indebted to my loving parents and in-laws for all the sacrifices they have made on my behalf. I would like to express sincere gratitude to my beloved wife Sanjana Dewangan who believed in me and provided encouragement during challenging times. Your unconditional love and support in the moments when there was no one to answer my queries has helped me immensely. To my son Shaurya and daughter Sanvi, who are the inspiration for me to complete this journey and also for the sacrifices made along the way. To all my friends thank you for your support and constant encouragement.

Contents

Declaration	i
Thesis Completion Certificate	ii
Thesis Completion Certificate from External	iii
Abstract	iv
Acknowledgment	vi
Contents	viii
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Introduction to Cloud Computing	1
1.2 Evolution of Cloud Computing	2
1.2.1 Cluster	2
1.2.2 Distributed Computing	3
1.2.3 High Performance Computing (HPC)	3
1.2.4 Application Service Provider (APA)	3
1.3 Cloud computing Service Models	5
1.3.1 Infrastructure-as-a-Service (IaaS)	5

1.3.2	Platform-as-a-Service (PaaS)	6
1.3.3	Software-as-a-Service (PaaS)	7
1.4	Deployment-Models	7
1.4.1	The Public Cloud	8
1.4.2	The Private Cloud	8
1.4.3	The Community Cloud	8
1.4.4	The Hybrid Cloud	8
1.5	Technology behind the Cloud Computing	9
1.5.1	Virtualization	9
1.5.2	Hypervisors	9
1.5.3	Multi-Tenancy	10
1.6	Cloud Computing Challenges	10
1.7	Resource Management	10
1.7.1	Resource Management in Cloud	10
1.7.2	Evolution of Resource Management	11
1.7.3	Why Resource Management?	12
1.8	Autonomic Resource-Management in Cloud	13
1.8.1	Self-management	13
1.8.2	Self-Configuration	14
1.8.3	Self-Healing	15
1.8.4	Self-Protection	15
1.8.5	Self-Optimization	16
1.9	Research Gaps	17
1.10	Metrics for Resource Management in Cloud	17
1.11	Research Motivation	19
1.12	Research Objective	19
1.12.1	Sub-Objectives	19
1.13	Organization of Thesis	19

1.14 Summary	21
2 Literature Review	22
2.1 Auction-Based Resource Management System (RMS)	23
2.2 Energy-Based RMS	25
2.3 Fault-Tolerant Based RMS	26
2.3.1 Reactive Frameworks	27
2.3.2 Proactive Frameworks	27
2.3.3 Frameworks by Industries	28
2.4 Nature and Bio Inspired RMS	29
2.5 Optimization-Based RMS	31
2.6 Cost-Based RMS	32
2.7 Profit-Based RMS	36
2.8 QoS-based RMS	37
2.9 Autonomic Cloud Resource Management	39
2.10 SLA-Based RMS	41
2.11 Research Challenges	44
2.12 Evaluation Parameters	45
2.13 Objective of the thesis	46
2.14 Summary	46
3 Methodology	48
3.1 The Proposed Research Methodology	48
3.2 Workload-Filter	50
3.2.1 Workload Dataset	50
3.2.2 Algorithm	50
3.2.3 User-Priority based on Execution Time	51
3.3 Self-optimization	51
3.4 Fault-tolerant	52

3.5	Resource Management	52
3.6	Performance Analysis	52
3.7	SMART-Architecture	52
3.8	Simulation & Test Environment	54
3.8.1	Simulation Environments	54
3.8.2	Cloudsim Simulation Toolkit	54
3.8.3	Amazon Web Services Environment	56
3.9	Summary	56
4	Self-optimization and Fault-tolerant Mechanism	58
4.1	Self-optimization	58
4.1.1	Formal Optimization Model	59
4.1.2	Modified ALO based Self-optimization	59
4.1.3	Modified ALO Operators	60
4.1.4	Random walk of ant (initial population/ solution)	61
4.1.5	Building Trap (Fitness Value)	62
4.1.6	Entrapment of ant in trap	64
4.1.7	Catching preys and rebuilding trap	65
4.1.8	Algorithm	65
4.2	Fault-tolerant Mechanism	66
4.2.1	Fault Detection Procedure	66
4.2.2	Threshold	67
4.2.3	Algorithm	68
4.3	Summary	69
5	Resource Management	70
5.1	Scheduling	70
5.2	Algorithm	71
5.2.1	Time Complexity	71

5.3	Results and Analysis	71
5.3.1	Performance Metrics	71
5.3.2	Validation of SMART	73
5.3.2.1	SOCCER	73
5.3.2.2	CHOPPER	73
5.3.3	Performance Analysis	74
5.3.3.1	Resource Utilization by Workload	74
5.3.3.2	Execution Time	74
5.3.3.3	SLA Violation Rate	74
5.3.3.4	Energy Consumption Rate	75
5.3.4	Execution Cost	75
5.3.5	Resource Cost	75
5.3.6	SLA Cost	76
5.3.7	Resource Utilization	76
5.3.8	Execution Time	77
5.3.9	Energy Consumption Rate	78
5.3.10	SLA Violation Rate Analysis	79
5.3.11	Cost	80
5.3.11.1	Execution Cost	80
5.3.11.2	Resource Cost	81
5.3.11.3	SLA Cost	82
5.3.12	Analysis	83
5.3.13	Implementation in AWS	84
5.4	Summary	87
6	Conclusions and Future Work	89
	References	94

List of Figures

1.1	Formal Cloud Computing model (OutrightSystems, 2019)	2
1.2	Evolution of Cloud Computing	3
1.3	Architecture of IaaS	5
1.4	Architecture of PaaS	6
1.5	Architecture of SaaS	7
1.6	Various categories of Resource Management in Cloud	12
1.7	Formal model of Self-Configuration for Cloud Resource Management	14
1.8	Formal model for Self-healing	15
1.9	Formal mode of Self-Protection for Cloud Computing	16
1.10	Formal model of Self-optimization for Cloud Computing	17
2.1	Usage rate of multi-objective functions under auction-based RMS .	24
2.2	Usage rate of multi-objective functions under energy-based RMS . .	26
2.3	Usage rate of multi-objective functions under fault-tolerant-based RMS	29
2.4	Usage rate of multi-objective functions under Nature-Bio-Inspired RMS	30
2.5	Usage rate of multi-objective functions under optimization-based RMS	32
2.6	Usage rate of multi-objective functions under cost-based RMS . . .	35
2.7	Usage rate of multi-objective functions under profit-based RMS . .	37
2.8	Usage rate of multi-objective functions under QoS-based RMS . . .	38
2.9	Usage rate of multi-objective functions under autonomic-based RMS	41

2.10	Distribution of various objective functions utilized in SLA-Based resource management in cloud computing	43
2.11	Evaluation Parameters Distribution cumulative	45
2.12	Evaluation Parameters Distribution for individual resource management techniques	46
3.1	SMART Methodology	49
3.2	Architecture of proposed research work SMART	53
4.1	Formal Optimization Model	59
4.2	Roulette wheel method (Mirjalili, 2015)	62
4.3	Fault VM computation.	67
5.1	Resource utilization based on workloads	76
5.2	Execution Time analysis based on number of workloads.	77
5.3	Energy Consumption Rate & Analysis based on Resources (VM).	78
5.4	SLA Violation Rate & Analysis based on workloads.	79
5.5	Execution Cost & Analysis based on workloads.	81
5.6	Resource Cost & Analysis based on workloads.	82
5.7	SLA Cost & Analysis based on workloads.	83
5.8	EC2 Dashboard	84
5.9	ALOScheduler EC2 Instance	84
5.10	Client application EC2 Instance	85
5.11	ALOScheduler, Client, and Worker EC2 Instance	85
5.12	Execution of workloads	86
5.13	Execution of workloads	86
5.14	Execution Time Analysis in AWS environment	87

List of Tables

2.1	Differentiation of Auction-based RMS by evaluation parameters. . .	23
2.2	Differentiation of Energy-based RMS by evaluation parameters. . .	25
2.3	Differentiation of fault-tolerant-based RMS by evaluation parameters.	28
2.4	Differentiation of nature and bio inspired RMS by evaluation pa- rameters.	30
2.5	Differentiation of optimized RMS by evaluation parameters.	31
2.6	Comparative study of Cost-based Resource Management in Cloud Computing.	33
2.7	Differentiation of profit-based RMS by evaluation parameters. . . .	36
2.8	Differentiation of QoS-based RMS by evaluation parameters.	38
2.9	Differentiation of autonomic cloud RMS by evaluation parameters. .	40
2.10	Differentiation of SLA-based RMS by evaluation parameters.	43
3.1	Dataset (Singh et al., 2016) (Gill et al., 2017)	50
3.2	Physical system configuration in which simulator installed.	55
3.3	Simulation environment details.	55
3.4	Configuration of simulation environment for CPU, RAM, and Band- width for PM_{max} and VM_{max}	56
3.5	Description EC2 Instance	57
5.1	Analysis	83

Chapter 1

Introduction

Overview

This chapter is all about the introduction to cloud computing, history, and evolution, the models and technologies which are utilizing in the cloud, various services offered by the cloud, cloud deployment models, and the issues and challenges are discussed. This research work aims to produce an efficient autonomic resource management technique in cloud computing, so besides, different resource management techniques, categories, evolution, motivation to opt research and the challenges are presenting with the research objectives of this work.

1.1 Introduction to Cloud Computing

Cloud computing is a platform where cloud users can access the resources (storage, servers, networks, application programs, etc.) over the Internet, either free of cost or on a rent basis which depends upon the service model that users opted by their own choice. Cloud users do not have control of underlying hardware infrastructure because they are owned and managed by the service provider ([Jayaswal & Shah, 2015](#)). As per NIST ([Mell & Grance, 2011](#)), there are five silent qualities of the cloud are:

1. On-Demand Self-Service,
2. Broad-Network Access,
3. Resource-Pooling,
4. Rapid-Elasticity, and

5. Measured-Service.

The formal cloud computing model is presented in FIGURE 1.1.

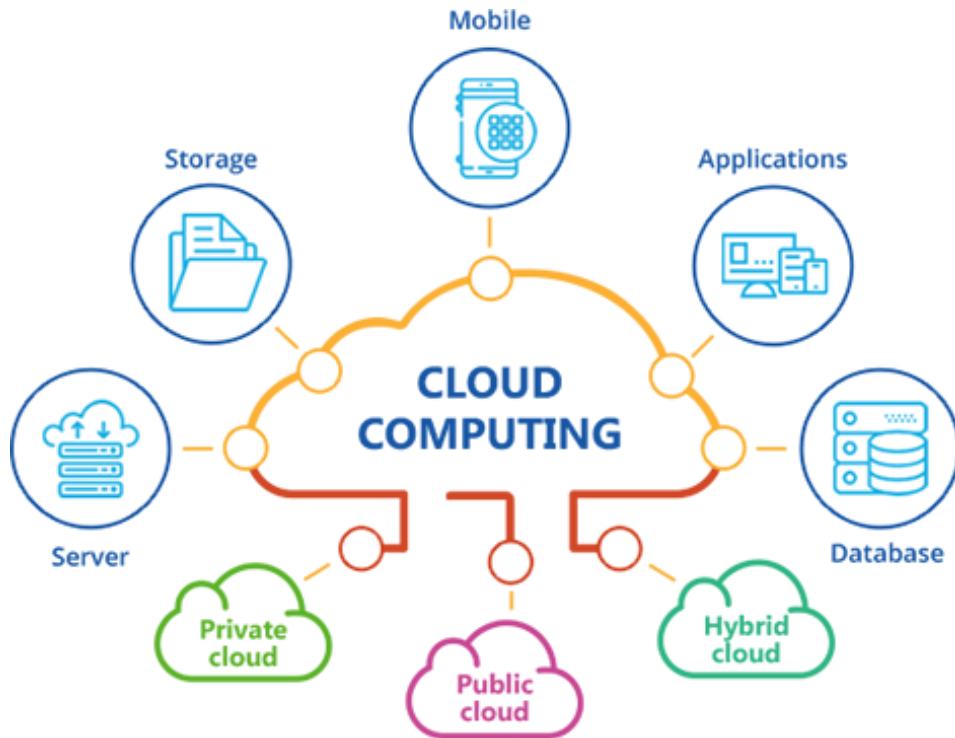


FIGURE 1.1: Formal Cloud Computing model (OutrightSystems, 2019)

1.2 Evolution of Cloud Computing

It is started by mainframe in the early 1960's, and the year-wise progress is shown in FIGURE 1.2. The mainframe was very expensive, even for large, profitable organizations. Hence, mainframe manufacturers provided a form of utility computing called time-sharing, where they offered database storage and compute power to banks and other large organizations for a fee.

1.2.1 Cluster

It is a group of networked systems sharing the same set of resources, where all the nodes are actively working or some nodes are in the standby mode, waiting to take over after the failure of an active node.

1.2.2 Distributed Computing

This is an implementation technique where different roles or tasks distributed among separate nodes in the network. Grid computing, peer-to-peer architecture, and client-server architecture are some forms of distributed computing.

1.2.3 High Performance Computing (HPC)

This technique divides a task into pieces and uses parallel processing algorithms to execute each piece on different processors on the same node or multiple nodes in the network.

1.2.4 Application Service Provider (APA)

An ASP in 1966 defined as an organization that hosts and manages one or more applications and its underlying infrastructure. Customers could use these applications over the internet and have to pay for their utilization.

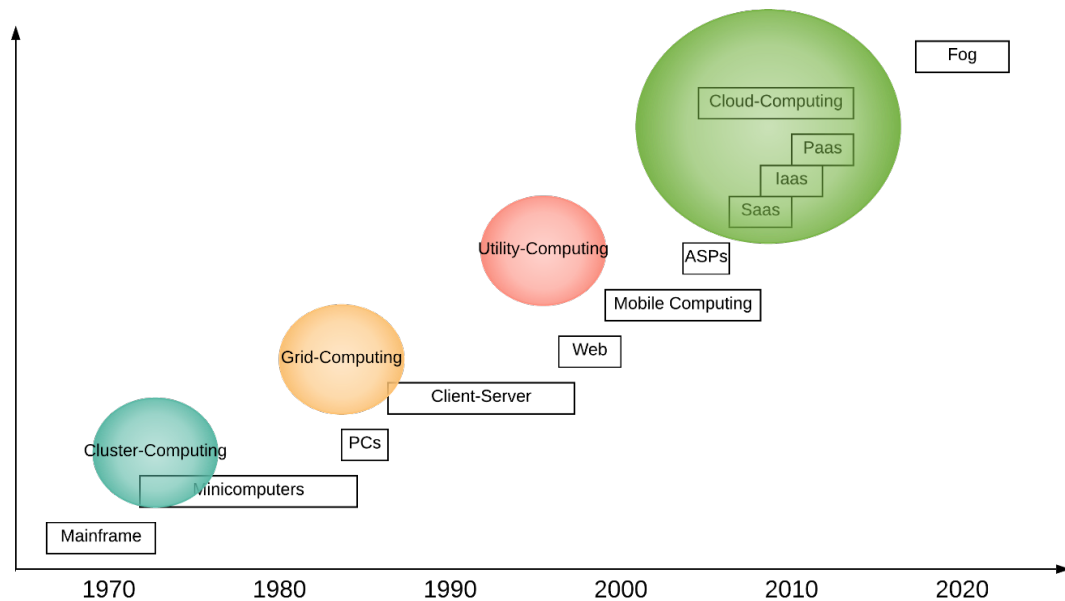


FIGURE 1.2: Evolution of Cloud Computing

In 1966, a firm named Computer Software System (CSS) started a time-sharing idea which is based on the control program/console monitoring system (CP/CMS).

In 1968, the concept of networked applications to support communities who could collaborate without regard to location introduces ([Licklider & Taylor, 1968](#)). The mainframe application is one of the examples of this concept.

In the 1970s and 80s, Digital Equipment Corporation (DEC) built and marketed one of the earliest time-sharing systems. During the 1990s and early 2000s, the internet used by application service providers for services that led to SaaS. Yahoo, Salesforce.com, and other internet pioneers provided cloud services several years before it named cloud computing.

In 1989, Tim Berner-Lee, a British computer scientist and MIT professor, created many web tools technical proposals that have today become the fundamental structure of the World-Wide-Web.

In 1990, with the help of Robert Cailliau and a student, carried out the first successful web-based communication over the internet between Hypertext Transfer Protocol (HTTP) server and client.

In the 1990s, utility computing re-surfaced, and in 1997, a company called InsynQ launched on-demand applications and desktop hosting services by using HP equipment. The following year, HP setup the Utility Computing Division in Mountain View, California and, launched its Utility Datacenter; marketing it as a service called IP-Billing-On-Tap.

In 2005, a company called Alexa launched the Alexa Web Search Platform, a web-based search-building tool with utility-type billing.

Since the 2000s, the primary form of cloud computing, namely, IaaS, PaaS, and SaaS, formalized.

In 2001, SIIA invented the acronym SaaS for a service that was the adoption of the Application Service Provider. The following year, Amazon started offering its infrastructure for web services for a pay-for-what-you-use model.

In 2006, 3tera launched its Applogic service, and later that summer, Amazon launched Amazon Elastic Compute Cloud (EC2) based on virtualized Linux and Windows servers. Both offered server and storage resources on a utility-based payment.

In November 2010, 11 companies, including Microsoft, Verizon, EMC, NetApp, Rackspace, Telenor, and Cisco Systems joined hands to form the Asia Cloud Computing Association. The global cloud computing market estimated to cross 70 billion US dollars by 2015.

1.3 Cloud computing Service Models

It provides three types of services:

1.3.1 Infrastructure-as-a-Service (IaaS)

In IaaS, a cloud user asks to pay for the resources stored at the service provider's infrastructure or wherever the provider keeps its hardware. The provider owns the equipment and maintains it at a level specified in the previously agreed upon Service Level Agreement (SLA). As a customer, they need to do is to pay for the part of resources dedicated permanently to customer account or resources that customer provision temporarily to meet the short term need. Rackspace-Servers, Amazon-EC2, Amazon-S3 are some vendors of IaaS.

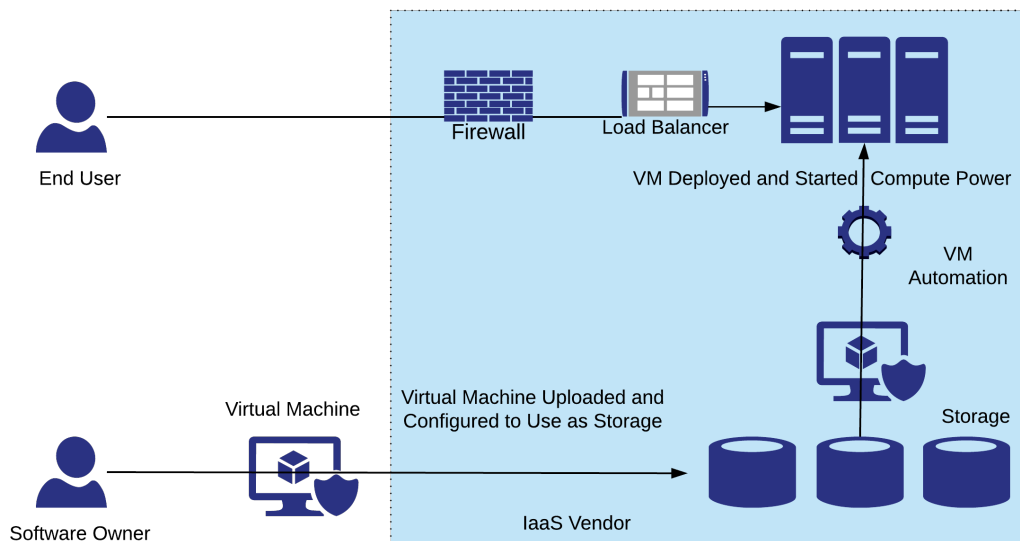


FIGURE 1.3: Architecture of IaaS

FIGURE 1.3, shows how a virtual appliance is constructed for as IaaS natural environment, uploaded, and configured and then established inside the environment. IaaS has two actors, that is, the service provider and an end-user. Software

programmer values virtual appliance and it is uploaded and configured to use the accessible storage. Software programmer also has command over virtual application automation where they are established and started. End-user submits a job or task to the virtual appliances, wherein the balancer splits up the task into a sub-task, submits it to the virtual applications, and obtains the results. The outcome is dispatched back to the end-user.

1.3.2 Platform-as-a-Service (PaaS)

In PaaS, a cloud user gets the set of applications and development environments like product development tools that are hosted on the service provider's hardware. In this, the user is allowed to write, compile, and deploy the code without installation of any tools on his computer. Users do not need to manage, control, upgrade those applications used by him. FIGURE 1.4 displays a PaaS model. PaaS has two actors, that is, developers and users. Developers evolve a stage comprising of IDE, security, supervising of application, and hardware infrastructure to evolve an application. End-user use the stage and establish their enterprise applications herein. Google-App-Engine, Microsoft-Azure, and Salesforce.com are comes in this category.

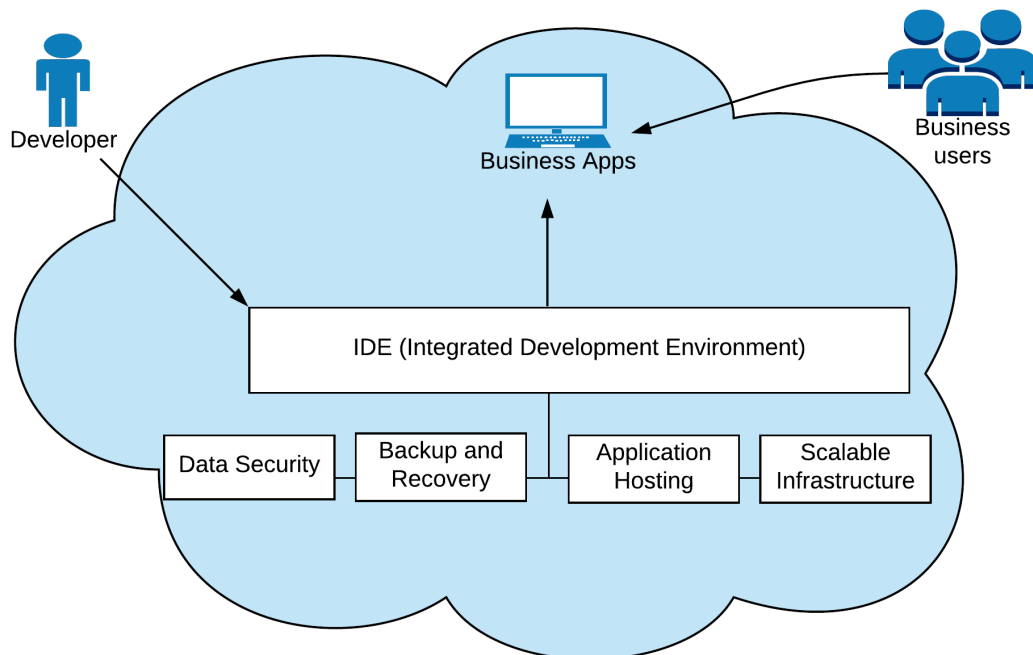


FIGURE 1.4: Architecture of PaaS

1.3.3 Software-as-a-Service (PaaS)

In SaaS, the application softwares intalled in cloud infrastructure is easy available to access by cloud user. User do not need to install the softwares in their computers. FIGURE 1.5, displays the diverse components accessible in SaaS. The components encompass metadata services, security services, and direct services. Security services are used for commanding access to end-user and back-end programs. Metadata services are used to organize application configuration for every tenant. Services and intelligence purchaser combine with the metadata services to get data that recounts configurations and additions that are apt for every tenant. FIGURE 1.5, displays the diverse components accessible in SaaS. The components encompass metadata services, security services, and direct services. Security services are used for commanding access to end-user and back-end programs. Metadata services are used to organize application configuration for every tenant. Services and intelligence purchaser combine with the metadata services to get data that recounts configurations and additions that are apt for every tenant.

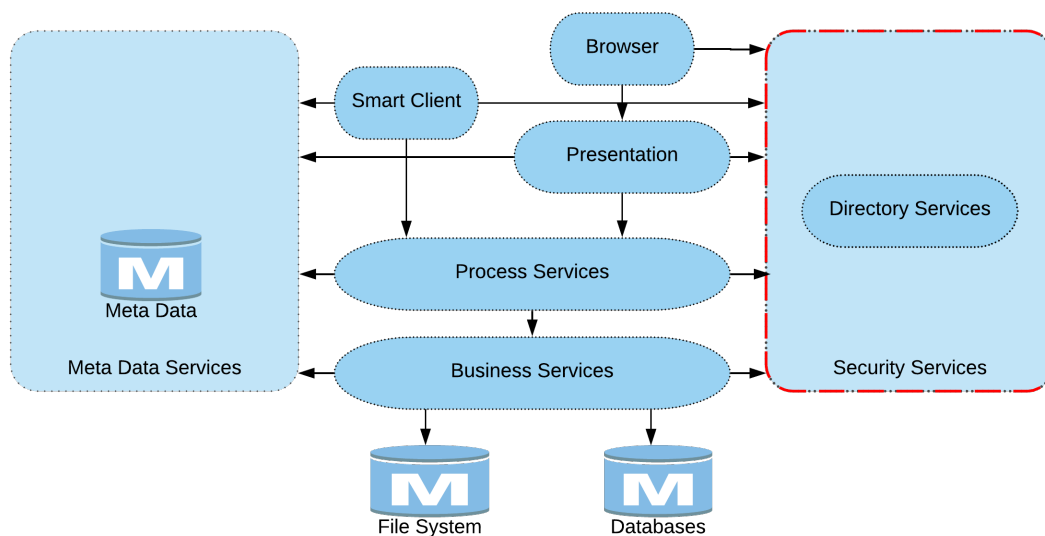


FIGURE 1.5: Architecture of SaaS

1.4 Deployment-Models

It is classified into four types as follows:

1.4.1 The Public Cloud

The clouds, accessed or used by general means and hosted, are maintained as well as managed by service providers. In this type of cloud, the service providers charge the companies according to their usage. Due to this, an initially small organization can start using cloud services and then can expand by acquiring more resources according to their requirements. During expansions, there is no need for the organization to invest in the infrastructure and can pay just according to usage. In the public cloud, there is no need for organizations to control or manage the resources; instead, they are being administered by a third party.

1.4.2 The Private Cloud

The cloud infrastructure is designed for a specific purpose or organization and it sharing with other organizations. As compared to a public cloud, private clouds are more costly as well as secure. A private cloud either can be on-premises or hosted externally. In the case of on-premise private clouds, the service is exclusively used and hosted by a single organization. However, the private clouds that are hosted externally are used by a single organization and are not shared with other organizations. Moreover, a third party that specializes in cloud infrastructure hosts cloud services.

1.4.3 The Community Cloud

In this, sharing is done between various organizations with usual tie. This type of cloud is generally managed by a third party offering the cloud service and can be made available on-premises or off-premises. To make the concept of community cloud clear and to explain when community clouds can be designed. In any state or country, the community cloud can be provided so that almost all government organizations of that state can share resources available on the cloud.

1.4.4 The Hybrid Cloud

The cloud environment in which various internal or external service providers provide services to many organizations is known as a hybrid cloud. Generally, it is observed that as organization host applications, which require a high level of security and are critical, on a private cloud. It is also possible that the applications requiring less of concern can be hosted on the public cloud. In hybrid clouds, an organization can use both types of cloud, i.e., public and private together. Such types of cloud are generally used in situations called as cloud bursting. In the

case of cloud bursting, an organization generally uses its computing infrastructure; however, in high load requirements, the organization can access clouds. In other words, the organization using a hybrid cloud can manage an internal private cloud for general usage and migrate the entire or a part of an application to the public cloud during the peak periods.

1.5 Technology behind the Cloud Computing

Virtualization, Hypervisors, and Multi-tenancy are the technology, which is used behind the cloud.

1.5.1 Virtualization

Virtualization is not a new idea for researchers. Memory was the first among the PC parts to be virtualized. Memory was a costly piece of the first computers, so virtual memory ideas were produced in the 1970s ([AlJahdali et al., 2014](#)). Virtualization technology separates the primary functions of computers, i.e., computing and technology implementation, from the physical infrastructure and the hardware resources with the help of a technology called virtual machine monitor (VMM). Virtualization changes the way businesses make their payments for using certain services, while risks associated with costs and payments for businesses are handled by it. The benefits associated with virtualization can be maximizing resources, reducing hardware costs, minimizing maintenance requirements, taking advantage of OS services, usage of multiple systems, and increasing the security of the system.

1.5.2 Hypervisors

Virtualized situations are typically imposed with the utilization of a Hypervisor, and it rest between the physical equipment and the Virtual Machines (VMs). Hypervisor responsible for furnishing each VM with the misconception of being kept running its equipment, which is finished by uncovering an arrangement of virtual hardware devices whose assignments are then planned on the genuine physical equipment. These administrations include some major disadvantages: Hypervisors are expansive bits of programming, with 100,000 lines of code or more ([Perez Botero et al., 2013](#)).

1.5.3 Multi-Tenancy

Multi-Tenancy is the naturalism of trying to attain the capital advantages in cloud computing by adopting virtualization technology via resource utilization (Jansen, 2011) and (Saripalli & Walters, 2010). Multi-Tenancy implies that at least two clients use a similar administration or application given by the CSP paying little mind to the fundamental assets resources. Multi-Tenancy happens when at least two virtual machines (VMs) having a place with various clients share the same physical machine (PM) (AlJahdali et al., 2014).

1.6 Cloud Computing Challenges

The broad area of issues of cloud computing is security, resource management, user trust gain, load balancing, costing model, and service level agreement (Dillon et al., 2010). Each broad area has its relative dimensions of areas, which are likely opted to resolve by the researchers. In this work, the resource-scheduling area is focused on including a multi-objective function to enhance resource utilization.

1.7 Resource Management

Resources are managed through its proper scheduling. Scheduling can be stated that an event to take place at a particular time. There are many types of scheduling algorithms available in distributed computing for resource scheduling. Many algorithms are to be utilized in the distributed system by appropriate authentication. The purpose of the scheduling algorithm is to achieve maximum throughput. For a cloud environment, the regular approaches are unable to attain the desired efficiency (Tanenbaum, 2009).

1.7.1 Resource Management in Cloud

Cloud computing classified the scheduling algorithms into the following categories; Batch/sequential and online/random mode. In batch/sequential mode, all the resources are standing in the chain and formed a set when it arrives at the system. In this, the algorithm will turn on in the fixed time interval. Cloud computing is an online technology and heterogeneous too so the speed of the processors can be made varied into less period, that's why the online mode of scheduling is more effective and suitable for the cloud (Dewangan et al., 2018).

1.7.2 Evolution of Resource Management

The evolution of resource planning parameters quality of service that are planned Resource-Scheduling-Algorithms (RSA) through the backstory of the cloud are described. any exceptional quality of service parameters (QoS) and focus of study (FOS) of resource planning for the evolution of the cloud in different years as outlined square in the evolution of resource planning as the next paragraph. As the cloud progresses with time and the introduction of innovative ways, so existing ASR within the cloud further progress. This study covers issues associated with the RSA support Quality of Service (QoS) and focus of study (FOS). Most work RSA cloud rising by reducing execution time, price and different quality of service parameters.

In 2009, ([Sotomayor et al., 2009](#)) and ([Buyya et al., 2009](#)) planned required virtual infrastructure hybrid formula mostly market and cloud-based bound based resource planning jointly, latency and execution time is taken into account as FoS QoS parameter is the virtual infrastructure and multiple distributed resources.

In 2010, ([Liu et al., 2010](#)), ([Kamal & Kemafor, 2010](#)), ([Lee et al., 2010](#)), and ([Pandey et al., 2010](#)) workflow provided bound compromised price and time based mainly point given time duration based mostly based service organization linked mainly load equalization linked primarily based VM advancement and application linked to nature and bio impressed formula jointly planning resources. Runtime, the profit rate, latency and resource utilization is considered as a parameter of quality of service and Fos is the cluster, and MapReduce exchange method.

In 2011, ([Yang et al., 2011](#)) and ([Linlin et al., 2011](#)) price mainly planned and based resource SLA algorithmic programming program jointly during which the parameters of quality of service are considered as violation of SLA, range, price and manageability resource. The aim of the study is the lack of uniformity of VM and user satisfaction.

In 2012, ([Bing et al., 2012](#)), ([Qiang, 2012](#)), and ([Changtian & Jiong, 2012](#)) conferred distributed QoS environment back home based primarily, optimizing return home to chance and DVS (Voltage scaling dynamic) based resource allocation particularly energy conscious solidarity algorithmic program. Time, resource utilization, energy consumption and utility system square measure thought about how quality of service parameters and Fos is the network performance, SLA, and consolidating workloads.

In 2013, ([Sotomayor et al., 2009](#)) planned consolidation priority back home mostly resources program programming algorithmic, during which measure square range and variety of migrations thought of as parameters of quality of service and Fos is workloads parallel.

In 2014, (Weiwei et al., 2014) and (Um et al., 2014) information back to severable task home awarded based mostly resource and CDN (Content Delivery Network) program algorithmic programming severally, in which the execution time, the price of resources, price transition is taken into account as quality of service parameters and Fos is separable programming tasks, resource consumption pattern and SLA.

1.7.3 Why Resource Management?

The main objective of resource management is to identify appropriate resources for planning workloads appropriate time and to extend the efficiency of resource utilization. In other words, the number of resources should be minimal for work to care for a necessary level of service quality, or reduce to the minimum the total time workload (or maximize performance) of a work. For greater resource planning, better allocation of labor resources needed. The second objective of resource planning is to identify the right job and appropriate supporting planning multiple workloads, they are able to meet diverse needs of service quality, such as using the hardware of the computer, manageability, reliability, safety, etc., for the cloud workload. Therefore, resource planning considers the execution time of each individual work, but most importantly, performance is also supported type of work that is, with totally different needs for quality of service (loads of heterogeneous work) and QoS requirements similar (homogeneous work loads).

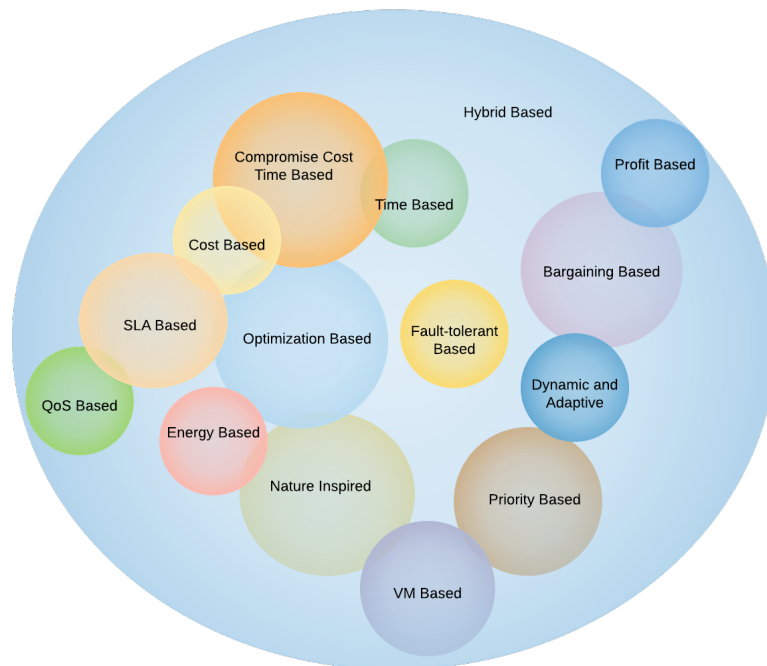


FIGURE 1.6: Various categories of Resource Management in Cloud

The above-mentioned resource management systems are presenting in FIGURE 1.6. The focus of this research is all about autonomic cloud resource management and dicussed in next section.

1.8 Autonomic Resource-Management in Cloud

It ¹ is self-managing characteristics of distributed computing resources that operates through policies. These systems are capable of self-healing, self-configuration of their resources, self-protection from malware, and attacks. An autonomic computing system maintains comprehensive knowledge of its components and the operating environment (self-knowledge) so that it can self-react to external inputs (self-adaption).To react automatically, these systems have built-in sensors that monitor the environmental conditions and external inputs to determine and execute the appropriate response actions (Thomas et al., 2013). The Autonomic Computing Initiative (ACI), which produced by IBM, exhibits, and backers organizing PC frameworks that do not include a considerable measure of human intercession other than characterizing input rules. The ACI got from the autonomic sensory system of the human body. Four territories of the scheduled is characterized by IBM through incorporate , self-healing, self-protection, self-configuration and self-optimization. Qualities that each autonomous processing framework must have automation, adaptively, and awareness. The various features of autonomic resources management in the cloud is discussed as follows:

1.8.1 Self-management

The fragrance of autonomic processing frameworks is self-management, an autonomic frameworks will keep up and modify their task despite evolving parts, outstanding burdens, requests, and outer conditions and notwithstanding equipment or programming disappointments, both guiltless and malevolent. The autonomic framework may persistently screen its very own utilization, and check for segment updates, for instance. On the off chance that it esteems the publicized highlights of the updates beneficial, the framework will introduce them, reconfigure itself as important, and run a relapse test to ensure everything is great. When it recognizes blunders, the framework will return to the more established adaptation while its programmed issue assurance calculations attempt to separate the wellspring of the mistake. IBM as often as possible refers to four parts of self-administration; early autonomic frameworks may regard these viewpoints as unmistakable, with various

¹ Autonomic Cloud Resource Management. *Fifth IEEE international Conference on Parallel, Distributed and Grid Computing(PDGC)*. PP: 138-143. 2018. DOI: <https://ieeexplore.ieee.org/document/8745977> DOI: 10.1109/PDGC.2018.8745977.Scopus Indexed.

item groups making arrangements that location every one independently. Eventually, these viewpoints will be new properties of a general design, and refinements will obscure into a more broad idea of self-upkeep. The voyage toward completely autonomic figuring will take numerous years; however, there are a few vital and profitable points of reference along the way. At first, computerized capacities will just gather and total data to help choices by human executives. Afterward, they will fill in as consultants, recommending conceivable strategies for people to consider. As robotization advances enhance, and our confidence in them develops, we will depend on autonomic frameworks with making and following up on lower-level choices. After some time, people should make moderately less regular overwhelmingly larger amount choices, which the framework will complete naturally through increasingly various, bring down level choices and activities.

At last, framework directors and end clients will underestimate the advantages of autonomic registering. Self-managing frameworks and gadgets will appear to be normal and unremarkable, as will computerize programming and middleware updates. The definite movement examples of uses or information will be as uninteresting to us as the subtle elements of directing a telephone call through the phone organize.

Self-manage technique includes the following capabilities:

1.8.2 Self-Configuration

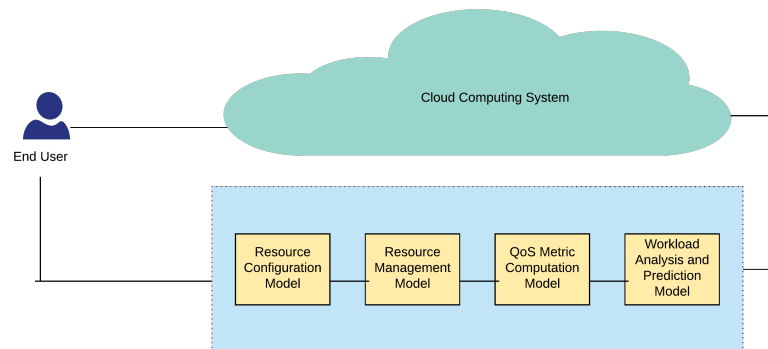


FIGURE 1.7: Formal model of Self-Configuration for Cloud Resource Management

Ability to accommodate varying and possibly unpredictable conditions. Introducing, designing, and incorporating expansive, complex frameworks is testing, tedious, and blunder inclined notwithstanding for specialists. Most huge Websites and corporate server farms are indiscriminate gradual additions of servers, switches, databases, and different advances on various stages from various sellers. It can take groups of master software engineers' a very long time to combine two

frameworks or to introduce a noteworthy online business application, for example, SAP. Autonomic frameworks will arrange themselves consequently as per abnormal state approaches speaking to business-level destinations, for instance - that determine what wanted, not how it is to be proficient (Omer et al., 2011). The formal model of self-configuration is presented in FIGURE 1.7.

1.8.3 Self-Healing

Ability to remain functioning when problems arise. IBM and other IT sellers have substantial divisions committed to recognizing, following, what is more, deciding the main driver of disappointments in complex processing frameworks. Genuine client issues can take groups of developers half a month to analyze and settle, and in some cases, the issue vanishes affectingly without any acceptable finding. Autonomic registering framework will recognize, analyze, what's more, limited improvement came about because of a bug problems or disappointments in programming and equipment, possibly through the analysis of relapse. Utilizing information about the design of the framework, is part of the problem determination (in light of the Bayesian system, for example) will examine the data from the log records, conceivably enhanced with information from the extra screen that has been requested. This framework will then match findings against known programming Repair (or alarm a software engineer humans if not present), introduced the match fixing, and retest. The formal model of self-healing is presented in FIGURE 1.8.

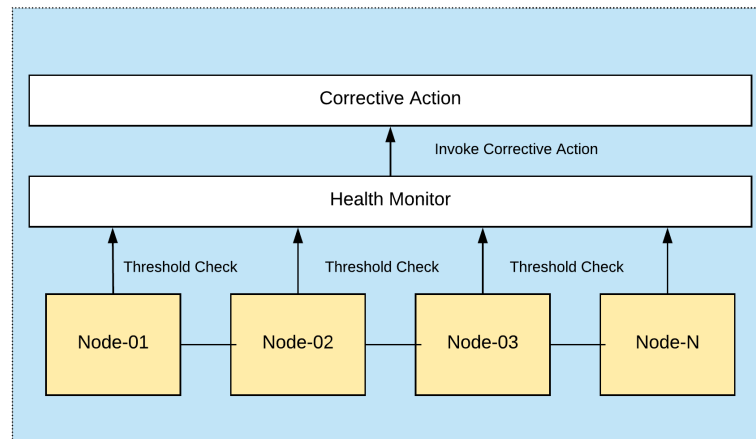


FIGURE 1.8: Formal model for Self-healing

1.8.4 Self-Protection

Ability to detect threats and take appropriate actions. Notwithstanding the presence of firewalls and interruption recognition apparatuses, people should at present

how to shield frameworks from pernicious assaults and coincidental falling disappointments. Autonomic frameworks will act naturally ensuring in two detects. They will protect the overall framework of the large-scale, related issues arise from malicious attacks or falls disappointment living corrected without any steps to recover the other. They will also envisage a problem depending on the initial reports of censorship and find ways to avoid them. The formal model of self-protection is presented in FIGURE 1.9.

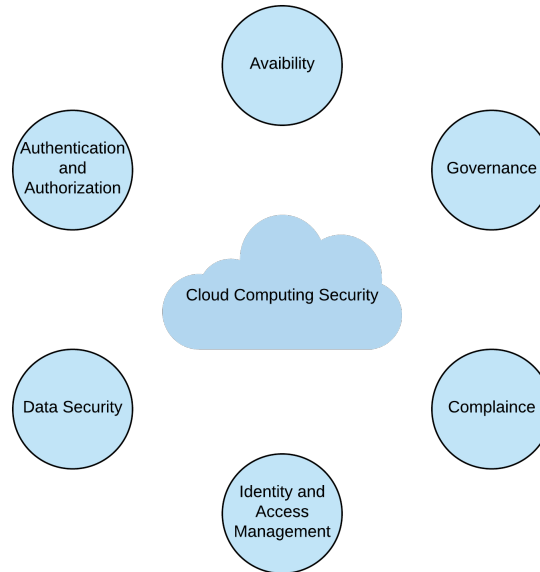


FIGURE 1.9: Formal mode of Self-Protection for Cloud Computing

1.8.5 Self-Optimization

Constant monitoring for optimal operation. Complex middleware, for example, Web Sphere, or database frameworks, Oracle or DB2, may have many tunable parameters that must be set effectively for the framework to perform ideally, yet few individuals know how to tune them. Such frameworks are frequently coordinated with other, similarly complex frameworks. Thus, execution tuning one substantial subsystem can affect the whole framework. Autonomous framework will be consistent looking approach to improving their duties, differentiate and seize the opportunity to make itself more effective in execution or fees. Similarly, as more grounded wind muscle through exercise, and brain adjust the hardware in the middle of learning, autonomous framework will display, explore different paths on and tune their parameters and will seek ways to settle on an appropriate decision about keeping capacity or distribute them. They will proactively look to remodel their capacity to seek, justify and apply the latest updates.

The formal method of self-optimization is presented in FIGURE 1.9.

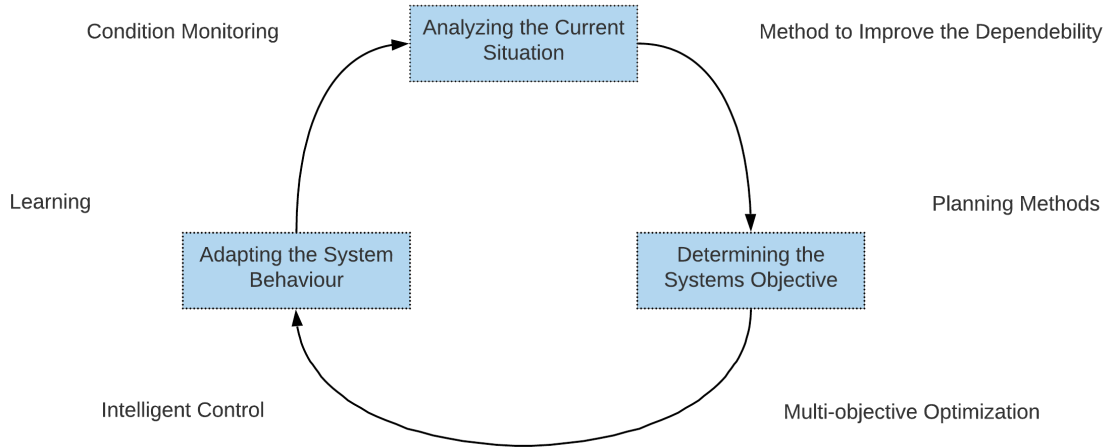


FIGURE 1.10: Formal model of Self-optimization for Cloud Computing

1.9 Research Gaps

Resource management plays a big role to minimize operating costs and maximize the efficiency of the system. Some of the following gaps identified in this research field:

1. SLA-violation-rate is high, which leads to less user trust,
2. Scheduling-time can be minimized,
3. Resource utilization is low,
4. Energy-consumption rate can be minimized,
5. Fault-tolerant for identifying fault virtual machines (VM's),
6. Resource cost can be optimized,
7. Response time is high,
8. Malicious workload identification is required,
9. Memory utilization rate is high,
10. More human interventions/ non-autonomic.

In this research, some of the above observations have been considered to make resource management efficient.

1.10 Metrics for Resource Management in Cloud

The following metrics used to measure resource management to evaluate the efficiency of the system:

1. **Resource Utilization:** The resource utilization (Gill et al., 2017) can be computed through following equation:

$$RU = \sum_{i=1}^m \frac{RC_{(i)} - ET_{(i)}}{RC_{(i)}} \quad (1.1)$$

Where RU is resource utilization by each workloads in ms, m is number of workloads, $RC_{(i)}$ is resource completion request, and ET is execution time.

2. **Execution Time:** The execution time for each workload is obtain through following equation which is used in Chopper (Gill et al., 2017):

$$ET = \sum_{i=1}^n \frac{RC_{(i)} - RS_{(i)}}{n} \quad (1.2)$$

Where $RC_{(i)}$ is request for workload completion time and $RS_{(i)}$ is the workload submission time, and n is number of workloads.

3. **Energy Consumption:** The energy consumption rate can be obtained through following equation, which is used in Soccer (Singh et al., 2016).

$$EnC = (EnC_{\max} - EnC_{\min}) * U_{VMi} + EnC_{\min} \quad (1.3)$$

Where EnC is energy consumption rate, EnC_{\max} is the energy consumption at the peak load (or 100% utilization), and EnC_{\min} is the minimum energy consumption in the active/idle mode (or as low as 1% utilization). In this research, EnC_{\max} is 1 kWh and EnC_{\min} is $0 < EnC_{\min} < 0.05$

4. **SLA Violation rate:** SLA violation rate is obtain through following equation:

$$SVR = failure_rate * \sum_{i=1}^n W_i \quad (1.4)$$

The above equations are used in CHOPPER (Gill et al., 2017) where, W_i is the weight of each workload.

5. **Average Cost:** The average cost can be computed through Execution cost, SLA cost and Resource cost, which is multiple of the cost and the above mentioned parameters.

1.11 Research Motivation

The use of cloud computing is increasing and becomes commercialized in small and big industries. Result in, more data centers and servers are installed and this process is never stopping process. As per the study conducted by (Anuj Kumar Yadav & Ritika, 2017) the energy consumption by servers are recorded as 53% which is highest as compared to power and cooling systems intalled in different data centers. The more servers, more electric power needs to running it continues. More power consumption results in more CO_2 emission, which leads to affect our environment. In addition to this, the large number of requests and its process needs an intelligent system to maximize efficiency while minimizing the power consumption, which can be opted by minimizing the execution time of resource provisioning and maximize its utilization.

1.12 Research Objective

To design a resource management system to improve the fault tolerance mechanism, self-adaptability, and maximize resource utilization in cloud computing.

1.12.1 Sub-Objectives

1. To attain the better system by incorporating an efficient and intelligent resource management algorithm,
2. To maximize resource-utilization through self-optimization,
3. To design Resource-Management-System (RMS) in cloud improves performance in terms of execution-time and average-cost through fault-tolerance by reducing the impact of failures on execution.

1.13 Organization of Thesis

The thesis is organizing in six chapters as follows:

Chapter 1

This chapter is all about the introduction to the research area from broad to narrow down to the research problem. It describes the various deployment and service models with issues and challenges. It describes the resource management and its issues, and the motivation to start work in this area. Based on the above study

the research objective defined and mentioned in this chapter, and at last, the organization of the thesis has discussed.

Chapter 2

The second chapter is all about the literature review. In this chapter, various cloud resource management is presenting with comparative study and outcomes of each literature. More than 200 articles from various reputed journals are access and near about 150 articles have been referred to this research.

Chapter 3

This chapter provides a brief idea about the various module of proposed research SMART. In this chapter, the research methodology has been discussed through research direction, and SMART architecture. The first module of SMART is the workload filter, which is described in detail. An algorithm is presented, which input is a list of normal workloads, and the output is filtered workloads, which assured to remove the redundant requests from cloud users.

Chapter 4

The second module of the proposed research work is self-optimization, which is presented in this chapter. The objective of this module is to find the optimal resource by adapting a modified antlion optimizer. The modified resource optimization technique has been proposed and presented through a multi-objective optimization algorithm. In continue to this, Fault-tolerant technique is also discussed in this chapter. The objective Fault-tolerant module is to identify the fault VM's based on the VM parameters (CPU, RAM, and Bandwidth) utilization. The fault-tolerant algorithm is discussed with its input and output. The input of the algorithm is list of VM's, and the out is list of best VM's.

Chapter 5

The resource management algorithm is devised in this chapter. The objective of this chapter is to propose a scheduling algorithm bu incorporation workload filter, self-optimization, and fault-tolerant technique. The input of the algorithm is a list of workloads and a list of VM's. This algorithm can schedule the workloads to best VM's, in minimum time without execution failure. SMART is simulated in CloudSim 3.0 and AWS EC2 instances. The proposed work is analyzed by the evaluation parameters, which are discussed in chapter 1 and chapter 2. And it is observed that SMART performs utmost as a comparison with existing works.

Chapter 6

This chapter the proposed research work SMART is concluded based on its area of research, challenges, methodologies used to overcome the research gaps with results and analysis. The limitations of SMART is discussed with future work.

Appendix A

This is Appendix A, the list of publications are included which is the outcome of this study.

1.14 Summary

In this chapter, cloud computing is introduced with its services and deployment models. The evolution is also discussed various technology used in the cloud. It also briefly described various challenges and issues in the cloud. The biggest challenge in the cloud is resource management is described and narrow down to the specific area of research is autonomic computing is focused and in continuous to this research gap is presented. Moreover, to overcome this research gap, the research objective has been designed, and to evaluate the results of this research work, the parameters are identified and discussed. The proposed research work SMART is also simulated in CloudSim 3.0 and also validated in a real-time cloud environment in AWS. The literature review of this research work is discussed in next chapter.

Chapter 2

Literature Review

Overview

In this, the well organized study of resource management strategies is explained. This study device RMS into different categories based on various parameters. In this investigation, approx. 250 articles from reputed journals/conference proceedings have been considered and out of these approx. 125 articles are reviewed and used. The considered research articles give an idea for testing of most significant RMS. The outcome of this study is based on the various parameters that are discussed through comparative tables and graphs. The different categories of Resource Management Systems RMS are identified as follows:

1. Auction-based
2. Energy-based
3. Fault-tolerant based
4. Nature and Bio Inspired
5. Optimization-based
6. Cost-based
7. Profit-based
8. QoS-based
9. Autonomic-based
10. SLA-based

The details discussion of the above mentioned categories of RMS are presented as follows:

2.1 Auction-Based Resource Management System (RMS)

Author Lin (Lin et al., 2010) addressed the peak or off-peak issues of resource allocation in the cloud and overcome it through the auction mechanism. The limit of this research is to develop a dynamic adapting technique which can overcome this problem more efficiently. Where author Salehi and Buyya (Salehi & Buyya, 2010) worked on the policies for resource allocation which increases the computational task of the local resource by adapting the resources from the cloud. The limit of this work is the absence of prior notification of application/workloads/task execution time. Author An B. discussed (An et al., 2010) about the dynamic negotiation mechanism, where service providers and end-user can negotiate the services through agents. This research has a delay in decision making and responsive service. Continue to this author Prodan (Prodan et al., 2011) proposed RMS, which is applied in scientific software in a distributed platform.

TABLE 2.1: Differentiation of Auction-based RMS by evaluation parameters.

Frameworks	SLA.	RU.	DuD.	Co.	ET.
(Lin et al., 2010)	✓	×	×	✓	×
(Salehi & Buyya, 2010)	×	✓	✓	✓	✓
(An et al., 2010)	×	✓	✓	✓	×
(Prodan et al., 2011)	×	×	×	✓	✓
(Zhangjun et al., 2013)	×	×	×	✓	✓
(Son & Jun, 2013)	✓	×	×	×	✓
(Xuejun et al., 2016)	×	×	×	✓	✓
(Ding et al., 2016)	×	✓	×	×	×
(Hong et al., 2016)	×	✓	×	✓	×
(Kong et al., 2016)	×	✓	✓	✓	×
(Xie et al., 2017)	×	✓	×	×	✓
(Wang et al., 2017)	×	×	×	✓	×
(Tafsiri & Yousefi, 2018)	×	✓	✓	×	×
(Jixian et al., 2018)	×	✓	×	×	✓
(Jin et al., 2018)	×	×	✓	✓	×

Abbreviation: ET= Executione-Time, SLA= Service Level Agreement Rate, Co=Cost, RU= Resource-Utilization, DuD= DueDate.

Author Zhangjun (Zhangjun et al., 2013) presented novel technique Task-To-VM which is based on a hybrid optimization algorithm, but not sure for a global optimal solution for Task-To-VM scheduling, where author Son (Son & Jun, 2013)

presented RMS which increases the SLA rate, throughput while minimizing failure rate. Author Xuejun (Xuejun et al., 2016) presented a research which is limited over response time, reliability, and user trust over the service provider. Authors (Ding et al., 2016) presented ABACUS RMS, author Hong (Hong et al., 2016) invent a novel technique COCA (Incentive-Compatible-Online-Cloud-Auction), which ensured user’s online demand based on user trust, moreover author Kong (Kong et al., 2016) discussed how auction RMS can be used for network transmission.

Author Xie (Xie et al., 2017) demonstrate that their proposed system accomplishes promising outcomes for virtual resource allotment, meanwhile Wang (Wang et al., 2017) and Tafiri (Tafiri & Yousefi, 2018) Presented RMS for distributed situations, where author Jixian (Jixian et al., 2018) proposed a heuristic-algorithm and authors Jin (Jin et al., 2018) propose an incentive-compatible-auction-mechanism which is used in resource replacements between the cloudlets and cell phones. The outcome of this study is compared based on evaluation parameters and presented in TABLE 2.1.

The results of the auction-based literature are shown in FIGURE 2.1. Where the cost objective function usage is 29.3%, which is the maximum usage of in auction-based literature in cloud resource management under cloud computing.

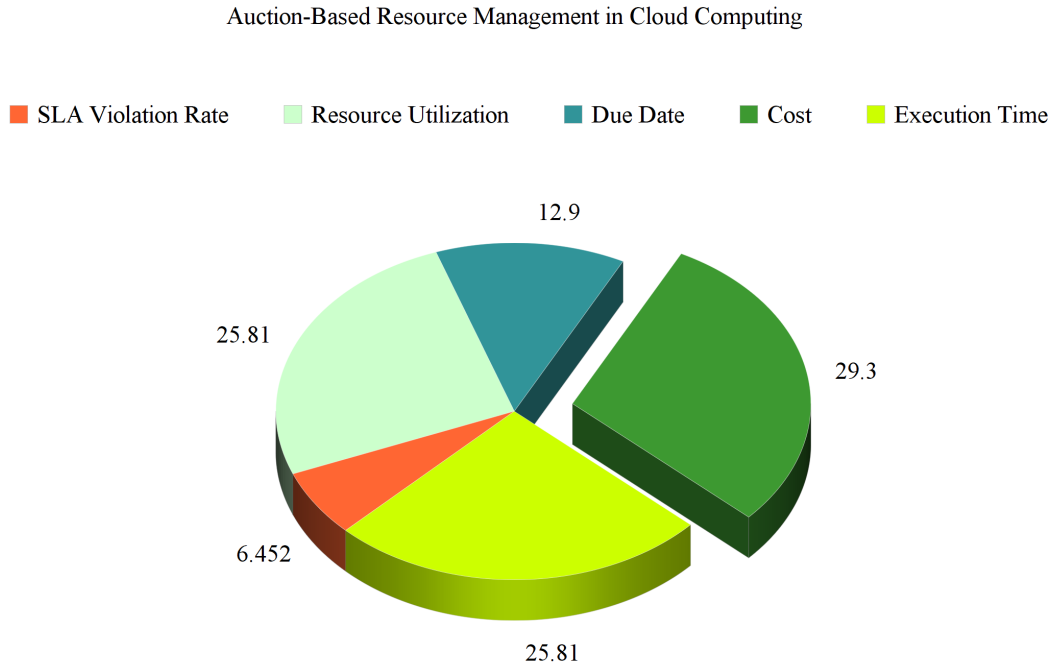


FIGURE 2.1: Usage rate of multi-objective functions under auction-based RMS

2.2 Energy-Based RMS

Author Tao (Tao et al., 2014) presented a makespan model to optimize the energy utilization based on hybrid optimization algorithm to consider as one of the streamlining goals from both financial and environmental points of view, where author L. X. Fang (L. X. Fang et al., 2014) proposed a nature-inspired algorithm to handle the virtual-machine placement VMP issues. A model for energy-aware resource usage procedure (Kansal & Chana, 2015) has been proposed to proficiently supervise cloud resources and upgrade their use, where author S. Guo (S. Guo et al., 2016) energy-based RMS to reduce power usage and compilation time.

TABLE 2.2: Differentiation of Energy-based RMS by evaluation parameters.

Frameworks	DuD	Cost	RU	Opt	PoC	QoS	ET
(Tao et al., 2014)	×	×	✓	✓	×	×	✓
(L. X. Fang et al., 2014)	×	×	✓	✓	×	✓	×
(Kansal & Chana, 2015)	×	×	✓	×	×	×	✓
(S. Guo et al., 2016)	✓	×	×	×	✓	×	×
(Gai et al., 2016)	✓	×	✓	×	×	✓	×
(Singh & Chana, 2016)	×	×	✓	×	×	✓	×
(Zhou et al., 2016)	✓	×	×	×	×	✓	×
(Su et al., 2013)	×	×	×	×	✓	✓	×
(Zhipiao et al., 2013)	×	×	✓	×	×	×	×
(Gang, 2014)	×	×	✓	×	×	✓	×
(Marzband et al., 2017)	×	✓	×	✓	×	×	✓
(Yibin et al., 2017)	✓	×	✓	×	✓	×	×
(Nir et al., 2018)	×	✓	✓	✓	×	×	×
(Tang et al., 2018)	✓	✓	✓	×	×	×	×
(Wen & Chang, 2018)	✓	✓	✓	×	×	×	×
(Alsadie et al., 2018)	×	×	✓	✓	✓	×	×

Where DuD=Due Date, RU=Resource Utilization, Opt=Optimization, PoC=Power consumption, QoS= Quality of Service, ET=Execution time.

The fundamental commitments of author Gai (Gai et al., 2016) are twofold. Initially, this paper is the principal investigation in taking care of energy squander issues inside the dynamic systems administration condition, where author Singh (Singh & Chana, 2016) presents energy-aware RMS which is based on fuzzy. Author Zhou (Zhou et al., 2016) presented energy-based RMS to increase QoS, where author Marzband (Marzband et al., 2017) is executed for load balancing offers. Another research by Yibin (Yibin et al., 2017), presented dynamic workloads allocation method to lower the power usage of mobile phones, where a new model

presented by Nir (Nir et al., 2018) demonstrated an optimal task allocation problem which is solved through by lowering power usage. Tang (Tang et al., 2018) proposed an energy-based RMS algorithm while considering information transmission. Continue to this, fair demand response with electric vehicles F-DREVs (Wen & Chang, 2018) presented a scheduling problem for cloud-based electrical vehicles arrangements, where Alsadie (Alsadie et al., 2018) presented a power usage based algorithm for Google cloud which decreases power consumption by 8.42%. The outcome of the study of energy-based RMS is compared based on evaluation parameters and presented in TABLE 2.2.

The results of the energy-based literature are shown in FIGURE 2.2. Where the resource utilization objective function usage is 30%, which is the maximum usage of in energy-based literature in cloud resource management under cloud computing.

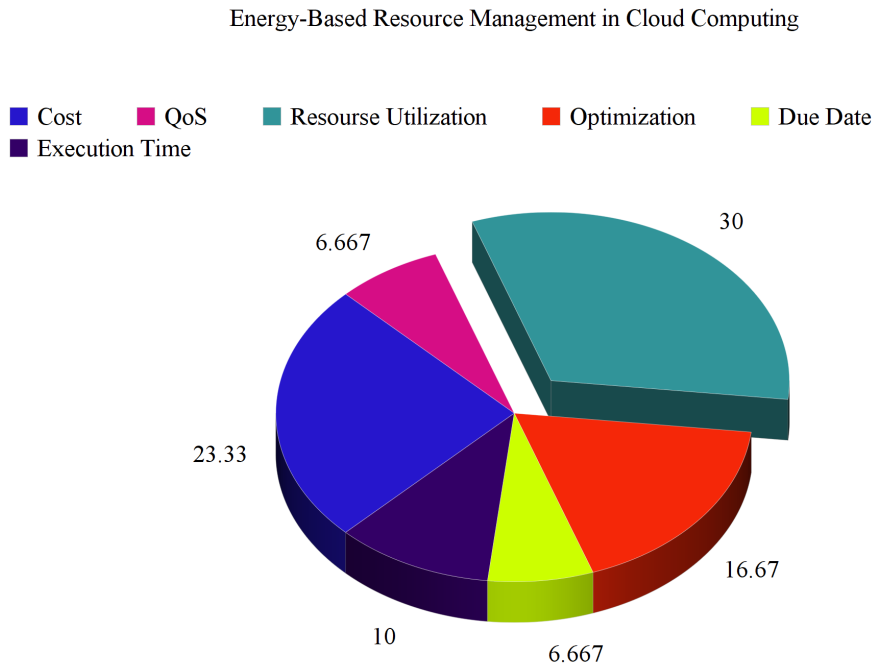


FIGURE 2.2: Usage rate of multi-objective functions under energy-based RMS

2.3 Fault-Tolerant Based RMS

Fault tolerance techniques can be further divide into two categories, Reactive and Proactive. Reactive can be further categorize into checkpoint restart, job migration

and replication as well as proactive also divided into preemptive migration, system rejuvenation as discussed in the following sub-sections:

2.3.1 Reactive Frameworks

Process Lever Redundancy (PLR) (Egwutuoha et al., 2012) presented a novel method for internal failure which leads to performance degradation in cloud computing, this method provides a fault-tolerant mechanism based on checkpoint/restart. Another research Fault-Tolerant-Mechanism (FTM) (Jhavar et al., 2012) presents an imaginative point of view on making and overseeing the adaptation of internal-failure. Continue to this, Latiff (Latiff et al., 2017) also presents a fault-tolerant model to handle the non-critical issue in a distributed computing environment. Malik (Malik & Huet, 2011) plan is profoundly to blame tolerant. The explanation for versatile unwavering quality is that the plan can exploit the dynamic adaptability of the cloud framework. Another side Multi-factor-monitoring-fault-tolerant (MRMFT) by Y. Fang (Y. Fang et al., 2018) worked for non-critical issues for Graphic Processing Unit GPU. In Deng's work (Deng et al., 2010) the research adaptation to non-critical failure and dependability issues in a more extensive scope of calculations. The issue of security assurance of the customer's information and results will be considered too. According to the survey by Nachiappan (Nachiappan et al., 2017) authors categorized various causes of failures like hardware, network, and software. A Low-latency-fault-tolerance (LLFT) (Wenbing et al., 2010) protect the system from various fault causes. Where Jayadivya (Jayadivya et al., 2012) and Poola (Poola et al., 2014) discuss the checkpoint/restart fault-tolerant model which lowering the cost and execution time as well while consideration of due date.

2.3.2 Proactive Frameworks

Shampio (Sampaio & Barbosa, 2018) gives a survey on adjustment to internal failures, specifically, proactive adjustment to internal failures, where Malik (Malik & Huet, 2011) works on the real-time virtualized model in which faults are identified proactively while maintaining the system reliability, where Zhang proposed FTCloud (Zheng et al., 2010) which identify faults automatically. Author Sun presented Fault-tolerant deadline-guaranteed (Sun et al., 2017) model which comprises four working parameters like a storm, diagram, equipment, and client space. Where Jing (Jing et al., 2015) offers a proactive model to avoid internal failures.

2.3.3 Frameworks by Industries

Google app engine throws the fault to the server automatically by app engine crone service where Giga spaces use the cluster techniques to divide the fault according to its type and that becomes a fault-tolerant cluster.

TABLE 2.3: Differentiation of fault-tolerant-based RMS by evaluation parameters.

Frameworks	SVR	QoS	ET	Cost	Eff	R	PoC
(Egwutuoha et al., 2012)	×	×	✓	×	×	×	×
(Jhawar et al., 2012)	×	✓	×	×	✓	×	×
(Latiff et al., 2017)	×	×	✓	×	×	×	×
(Malik & Huet, 2011)	×	✓	×	×	✓	×	×
(Zheng et al., 2010)	✓	×	✓	✓	×	×	✓
(Sun et al., 2017)	×	×	✓	✓	✓	✓	×
(Jing et al., 2015)	✓	×	×	×	×	✓	×
(Rimal et al., 2009)	×	×	×	×	✓	✓	×
(Y. Fang et al., 2018)	×	×	✓	×	×	×	×
(Deng et al., 2010)	×	✓	×	×	✓	×	×
(Nachiappan et al., 2017)	×	×	✓	×	×	×	×
(Wenbing et al., 2010)	×	✓	×	×	✓	×	×
(Jayadivya et al., 2012)	✓	×	✓	✓	×	×	✓
(Poola et al., 2014)	×	×	✓	✓	✓	✓	×
(Sampaio & Barbosa, 2018)	✓	×	×	×	×	✓	×

Where SVR= SLA violation rate, QoS= Quality of Service, ET=Execution time, Eff= Efficiency, R= Reliability, PoC= Power consumption.

Microsoft Azure used load-balancing technique according to its availability if any fault arises then the standardized query language (SQL), data service active automatically to resolve the issue by using another replica of the container, which is a reactive method. Another service provider Right Scale handle failure rate by advance architecture using elastic internet protocols (IP's), and force.com using self-management and self-healing techniques to solve the failure rate (Rimal et al., 2009) and that is type of proactive fault tolerance method. It identifies the faulty resources automatically through threshold value and the resources below this value, are separate from the resource pool and available resources are assign to workload according to user priority. The outcome of the study of fault-tolerant-based RMS is compared based on evaluation parameters and presented in TABLE 2.3.

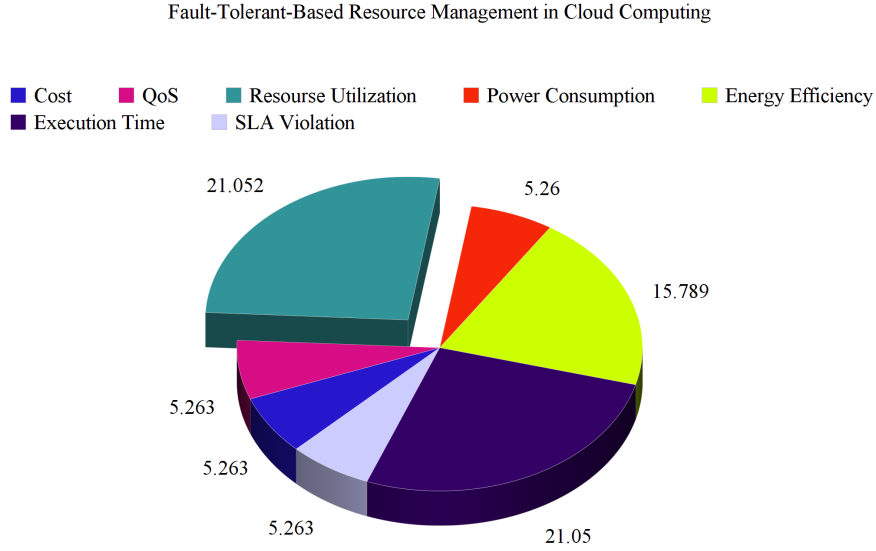


FIGURE 2.3: Usage rate of multi-objective functions under fault-tolerant-based RMS

The results of the fault-tolerant-based literature are shown in FIGURE 2.3. Where the resource utilization and execution time objective function usage is 21.05%, which is the maximum usage of in fault-tolerant-based literature in cloud resource management under cloud computing.

2.4 Nature and Bio Inspired RMS

Author Navimipour presented a novel algorithm, (Navimipour, 2015) to plan the errands in Cloud registering and Abdullahi (Abdullahi et al., 2016) discuss Symbiotic algorithm for cloud resources in a static environment. Author Kaur and Mehta (Kaur & Mehta, 2017) proposed Frog Leaping based method for resource provisioning in IaaS. Author Midya (Midya et al., 2018) presented a three-tier architecture, which uses vehicular-cloud, roadside-cloudlet, and unified-cloud while maintaining the QoS. Once again Mehta and Kaur (Mehta & Kaur, 2019) presented Nature-Inspired-Algorithms which is hybrid of Particle-Swarm-Optimization, Grey-Wolf-Optimization Algorithm and Shuffled-Frog-Leaping-Algorithm for cloud resource allocation. The results of the nature-bio-inspired literature are shown in FIGURE 2.4. Where the execution time and optimization objective function usage is 33.33%, which is the maximum usage of in nature-bio-inspired literature in cloud resource management under cloud computing.

Nature-Bio-Inspired Resource Management in Cloud Computing

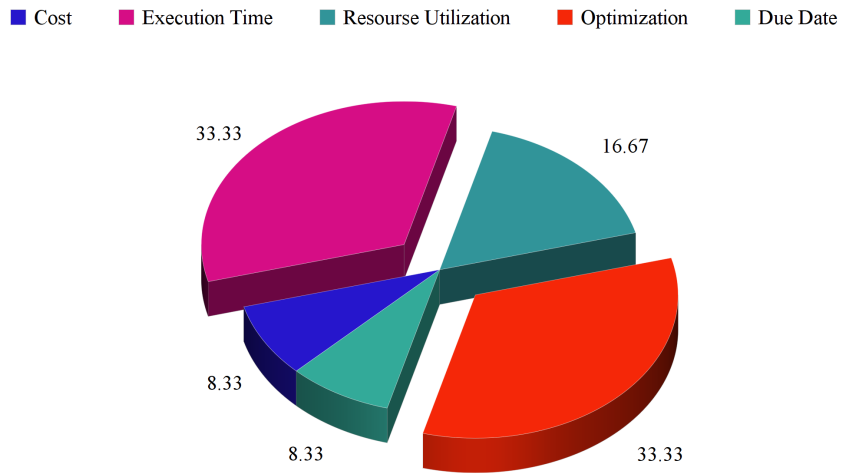


FIGURE 2.4: Usage rate of multi-objective functions under Nature-Bio-Inspired RMS

The outcome of the study of nature and bio inspired RMS is compared based on evaluation parameters and presented in TABLE 2.4.

TABLE 2.4: Differentiation of nature and bio inspired RMS by evaluation parameters.

Frameworks	DuD	Cost	RU	Opt	ET
(Navimipour, 2015)	×	×	×	✓	✓
(Abdullahi et al., 2016)	×	×	✓	✓	✓
(Kaur & Mehta, 2017)	✓	×	×	×	✓
(Midya et al., 2018)	×	×	✓	✓	✓
(Mehta & Kaur, 2019)	✓	✓	✓	×	×

Where DuD= Due date, RU= Resource utilization, Opt= Optimization, ET=Execution time.

2.5 Optimization-Based RMS

Particle-Swarm Optimization-Algorithm (C. Guo & Yu, 2005) is proposed by C. Guo, which is a self adaptive model by using of swarms properties, where Genetic-Algorithm (Sivanandam & Deepa, 2008) works for a critical framework. GA is commonly known for easy handling of cloud resource allocation issues. The nature-inspired optimization algorithm like Ant-Colony-Optimization-Algorithm (Al Salami, 2009) proposed by Al Salami, which uses a random selection technique, Continue to this a hyper-heuristic model (Das et al., 2009) is investigated by Das. Another algorithm based on GA is CCGA proposed by Yusoh (Yusoh & Tang, 2012) in 2012, which is an alternative solution for GA. Author Chaisiri (Chaisiri et al., 2012) investigated novel heuristic algorithm for reservation of the closest neighbor's to connect the resource demand. In advance Coutinho (Coutinho et al., 2013) control the memory and processor usage of servers. Author Banu and Saravanan proposed Short-Term-Planning-Algorithm (Banu & Saravanan, 2014) which optimizes cost and subscription policy. Authors Choi and Lim presented Auction-Based-Resource-Co-Allocation Algorithm (Choi & Lim, 2016) which maximizes resource cost and utilization , where Soccer (Singh et al., 2016) is an algorithm to focused on QoS. Author Qiu presented RPE Optimization-Technique (Qiu et al., 2017) which is used to load balancing for physical servers. Author Alex (Alex et al., 2017) lowering the power consumption. The outcome of the study of optimized RMS is compared based on evaluation parameters and presented in TABLE 2.5.

TABLE 2.5: Differentiation of optimized RMS by evaluation parameters.

Frameworks	SVR	QoS	ET	Cost	Enrg	PoC	CO2
(C. Guo & Yu, 2005)	×	×	✓	✓	×	×	×
(Sivanandam & Deepa, 2008)	×	✓	×	×	✓	×	×
(Al Salami, 2009)	×	✓	×	×	×	×	×
(Das et al., 2009)	×	×	✓	y	×	×	×
(Yusoh & Tang, 2012)	×	×	×	✓	×	×	×
(Chaisiri et al., 2012)	×	×	×	✓	×	×	×
(Coutinho et al., 2013)	×	×	y	✓	✓	×	×
(Banu & Saravanan, 2014)	×	✓	×	×	✓	×	×
(Choi & Lim, 2016)	✓	✓	✓	✓	×	×	×
(Singh et al., 2016)	✓	✓	✓	✓	✓	×	×
(Alex et al., 2017)	×	×	×	✓	✓	✓	✓
(Qiu et al., 2017)	×	×	×	×	×	✓	×

Where SVR= SLA violation rate, QoS= Quality of Service, ET=Execution time, Enrg= Energy, PoC= Power consumption.

The results of the optimization-based literature are shown in FIGURE 2.5. Where the cost objective function usage is 33.33%, which is the maximum usage of in optimization-based literature in cloud resource management under cloud computing.

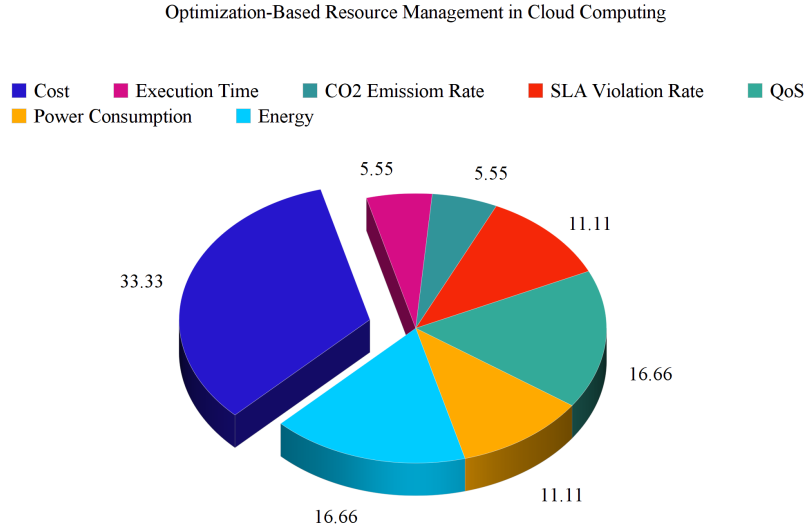


FIGURE 2.5: Usage rate of multi-objective functions under optimization-based RMS

2.6 Cost-Based RMS

We have surveyed of different resource management technique in terms of cost as (Teng & Magoules, 2010) proposed resource planning procedures for estimating costs on the balance of assets prospects without knowing the competitor's next victim Nash equilibrium data and tasks so far obtained by customers and meet deadlines and cost constraints by simulating in CloudSim. (Liu et al., 2010) Recommended time bid cost of the asset based on reservations and take time execution and cost QoS parameters. This methodology meets the client planned maturity date and achieve lower the burden of all instances of work processes while but does not consider heterogeneous. (Oprescu & Kielmann, 2010) Proposes spending calculation imperative asset for booking assignments bag in which the company is chosen to light first come, first be served technique. This instrument limit cost, time and performance improves processor performance however; because of the problem of hunger, this system is not convincing. (Van den Bossche et al., 2010) discussed the problem of improving conditions such as execution of the work force

is not preemptible date and as required under conditions of cross breed cloud multivendor light of the needs of information transfer, memory and CPU load remains migrateable completed. (Moschakis & Karatza, 2011) Published VM resource-based strategic planning to evaluate the total cost of the reservation Gang carefully hunger and relocation decision and execution of elite business applications. To manage hunger in the booking system, organized lines used to find the needs of each application in the light of their maturity date and other coveted.

TABLE 2.6: Comparative study of Cost-based Resource Management in Cloud Computing.

Frameworks	SVR	QoS	RU	Opt	DuD	ET
(Liu et al., 2010)	×	y	×	×	×	✓
(Teng & Magoules, 2010)	×	×	✓	×	×	×
(Van den Bossche et al., 2010)	×	×	×	×	✓	×
(Oprescu & Kielmann, 2010)	×	×	×	×	×	✓
(Bittencourt & Madeira, 2011)	×	✓	✓	✓	×	✓
(Moschakis & Karatza, 2011)	×	×	✓	×	×	✓
(Iyer & Veeravalli, 2011)	×	×	×	×	✓	✓
(Zhipiao et al., 2013)	✓	×	×	×	×	×
(Su et al., 2013)	×	×	×	×	✓	✓
(Altmann & Kashef, 2014)	×	×	✓	×	×	×
(Gang, 2014)	×	×	✓	×	×	✓
(Kang et al., 2014)	✓	×	y	×	✓	×
(Bansal et al., 2015)	×	✓	×	×	×	×
(Malawski et al., 2015)	×	×	✓	×	✓	✓
(Hoenisch et al., 2015)	×	×	✓	×	×	×
(Gan et al., 2015)	×	×	✓	×	×	✓
(Meena et al., 2016)	×	×	✓	×	✓	✓
(Zhongjin et al., 2016)	×	×	×	✓	✓	✓
(Koch et al., 2016)	×	×	✓	×	×	×
(Convolbo & Chou, 2016)	✓	×	×	×	×	×
(Zuo et al., 2017)	×	×	✓	✓	✓	✓
(Ghasemi et al., 2017)	×	×	×	✓	×	✓
(Alkhanak & Lee, 2018)	×	×	×	×	×	✓
(Sahni & Vidyarthi, 2018)	×	×	×	×	✓	✓

Where QoS= Quality of Service, SVR=SLA violation rate, Opt=Optimization, DuD=DueDate, RU=Resource Utilization, ET=Execution time, and C=Cost.

(Bittencourt & Madeira, 2011) Optimization techniques presented charges based planning hybrid system to take care of the need for active execute work process

within the spending plans and dynamic display performance using virtual machines to improve research on assets (with finding sufficient assets to light preconditions QoS). (Iyer & Veeravalli, 2011) Presented evaluate asset calculations on the reservations, two indisputable dealing engineering work processes free. Bargaining techniques can handle continuous occupation affirmation and elements of work despite assurances texts appropriate bargaining fairness. (Su et al., 2013) proposed a heuristic algorithm, with low costs production margin. This procedure focuses on expending for non-business-related to the primary application.

(Zhipiao et al., 2013) SLA proposes hereditary calculation based component bookings conscious asset is satisfactory by the virtual machine to produce the outer runway on the lease. Author (Gang, 2014) propose a change in the light of the calculation of the PSO to take over business problems booking in situations of cloud computing. In particular through the conscious inclusion of health costs ability to measure the cost of resource utilization, in addition to health work for the cost of time, the strategy author can achieve the goal of limiting both prepare time and resources are being used, and in this way achieve that ideal all over the world results. In this paper, authors (Altmann & Kashef, 2014) propose a model for the kind of broad-based cost of the cloud, be certain hybrid clouds. The author shows how the calculation of the cost module and administrative situation in a private cloud situation. Their results show that the algorithm administrative situation with the cost of the event to limit spending for computing services.

Author Kang (Kang et al., 2014) propose a Heuristic workflow plan based programming model considering cloud-judge in this document. Its plan consists of two stages: pressing and VM (multiple resource requests Single) MRSR stage. Author Malawski (Malawski et al., 2015) checking, creating and evaluating new algorithms in light of static and dynamic techniques for schedules and purchases. their results show that the strategy of discernment in the light of the structure of the work process and gauges of business altogether run-time can improve the setting properties. In (Bansal et al., 2015), the cost is included computing tasks of the quality of service based and contrast planning and resource scheduling algorithms in the cloud environment. The test results in the light cloudsims3.0 toolbox with NetBeans IDE8.0 shows that the finish quality-oriented high-performance service in the cost parameters. (Gan et al., 2015) Propose a dynamic destination technique is also offered to provide a genetic algorithm GA chance to center around improving execution time of order to meet the deadline is important when arranging brands feels not received.

Authors (Hoenisch et al., 2015) proposes eBPMS is to enhance the productivity of process institution, specifically with respect to adaptability and cost-proficiency. In (Zhongjin et al., 2016) proposed an technique depends on the meta-heuristic optimization methods, PSO, coding techniques have been designed to minimize implementation costs of overall work process while respecting the dates and the level of risk because of the imperative. Authors (Meena et al., 2016) proposed a

profitable meta-heuristic genetic algorithm that limits the cost of implementing work processes while fulfilling the due date on a distributed computing environment. Make a new plan for encoding, initialization of the population, crossover and mutation operator of the genetic algorithm. Authors (Convolbo & Chou, 2016) Target to propose and take over the costs for programming progress DAG on cloud IaaS stage where planning tasks must adapt to the procurement of assets to achieve the ideal setting. In this paper, they propose both ideal and calculation heuristic reservations, and rated them more DAGs utilizing various display value of EC2. Writer (Koch et al., 2016) presents techniques based on maximum likelihood estimates assume heterogeneous IT framework with the ultimate goal of designing resource allocation design that extends the use of the stage for the situation instructive.

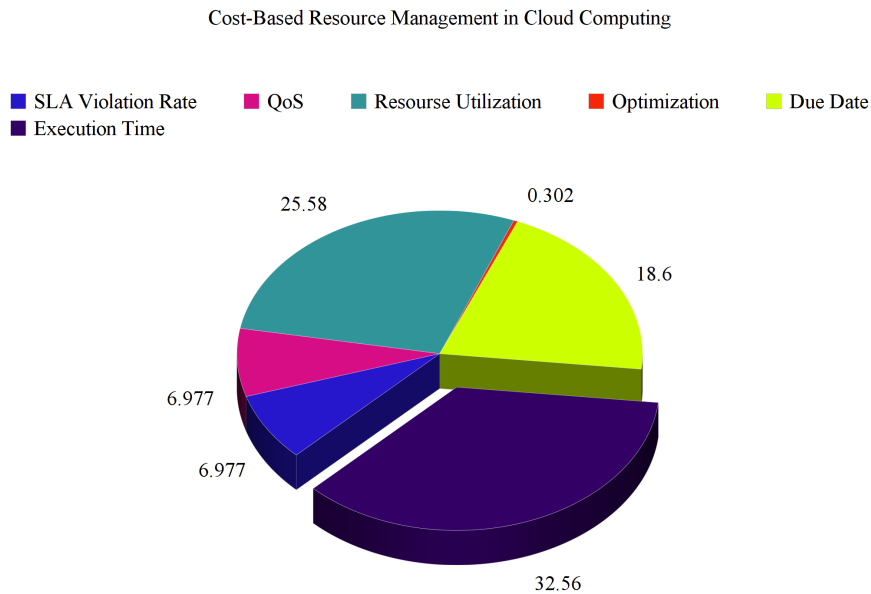


FIGURE 2.6: Usage rate of multi-objective functions under cost-based RMS

(Ghasemi et al., 2017) based approach to resource Proposed new procurement study completed certification of cost-reduction in demand. Commitment provision (ORP) Optimized Resource approach us. Writer (Zuo et al., 2017) propose a model that focuses Multi-Objective scheduling in the light of nature content optimization algorithm in distributed condition of computation as shown by the due date and the imperative of cost.

This work (Sahni & Vidyarthi, 2018) propose cost-effective deadline-constrained dynamic heuristic calculation to logically plan work processes in the open cloud. (Alkhanak & Lee, 2018) Propose completion time of hyper-heuristic driven methodology for the cost of progress. Provide a broad foundation in the planning process

of logical work to the cloud. The proposed approach helps in saving cost and time of cloud specialist organizations. The comparison made of various related research which is shown in TABLE 2.6.

The results of the cost-based literature are shown in FIGURE 2.6. Where the execution time objective function usage is 32.56%, which is the maximum usage of in cost-based literature in cloud resource management under cloud computing.

2.7 Profit-Based RMS

Writer Mei (Mei et al., 2017) presented a model which is focused on consumer loyalty. Consumer loyalty affect the interests of co-ops specialists in many ways. First, affect cloud design administrative nature, which is an important factor affecting consumer loyalty. Second, in entry-levelthe loyalty affect. This paper (Reddy et al., 2017) defines bookings pleasant floral problems vitality showcase of wind and sun by Independent Power Producers (IPP) on both traditional substances and Renewable Energy Technologies rets as distinguished. Writer Gao (Gao et al., 2017) proposed system enhancements stochastic two-timescale to strengthen administrative purposes while achieving prerequisite execution together with the provision of resources and tasks allocation. The writer Yuan (Yuan et al., 2017) proposed Hybrid Heuristic Optimization Algorithm, reproduced Particle Swarm Optimization (PSO), explained the sub-problems in each cycle of PMA , Additionally, PSO contrast and benchmark existing calculations.The comparative study of profit-based approach is shown in TABLE 2.7.

TABLE 2.7: Differentiation of profit-based RMS by evaluation parameters.

Frameworks	Enrg	Cost	RU	Opt
(Gao et al., 2017)	×	✓	✓	×
(Mei et al., 2017)	✓	✓	×	×
(Yuan et al., 2017)	✓	✓	×	✓
(Reddy et al., 2017)	✓	✓	✓	×

Where Enrg= Enrgy, RU= Resource utilization, Opt= Optimization.

The results of the profit-based literature are shown in FIGURE 2.7. Where the cost objective function usage is 40%, which is the maximum usage of in profit-based literature in cloud resource management under cloud computing.

Profit-Based Resource Management in Cloud Computing

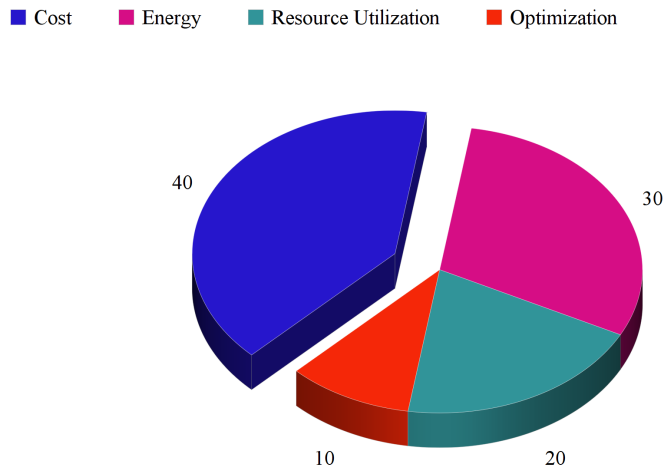


FIGURE 2.7: Usage rate of multi-objective functions under profit-based RMS

2.8 QoS-based RMS

Author Delimitrou ([Delimitrou & Kozyrakis, 2014](#)) present Quasar, administrative framework that extends the resource group to use when giving a high reliability application execution. In this paper the authors Dai ([Dai et al., 2015](#)) gifts, novel task-GAACs MQoS calculation reservations with the proposed multi-QoS constraints, given the boring, use, safety and reliability in planning procedures. This calculation incorporates the calculation of ACO and GA, where the author Sandholm ([Sandholm et al., 2015](#)) offers Other means QoS level as far as time fulfilling their work and present another offering component-based QoS to the work group in the group OpenStack multi-residents. Author Singh ([Singh & Chana, 2015](#)) reward system administration tasks cloud productive left where cloud incredible load has been recognized, damaged and broken through the K-means based on the weight lifted and QoS they need, where authors Dou ([Dou et al., 2017](#)) propose VM dynamic scheduling techniques to improve QoS energy conscious in the fog on a huge update to address the above test.

TABLE 2.8: Differentiation of QoS-based RMS by evaluation parameters.

Frameworks	Enrg	Cost	RU	Opt	ET
(Delimitrou & Kozyrakis, 2014)	✓	✓	×	×	×
(Dai et al., 2015)	×	✓	✓	×	×
(Sandholm et al., 2015)	×	×	✓	×	✓
(Singh & Chana, 2015)	✓	✓	✓	✓	×
(Dou et al., 2017)	×	×	✓	×	×
(Jala & Ramchand, 2019)	×	×	✓	×	✓

Where Enrg= Energy, RU= Resource utilization, Opt= Optimization, ET= Execution time.

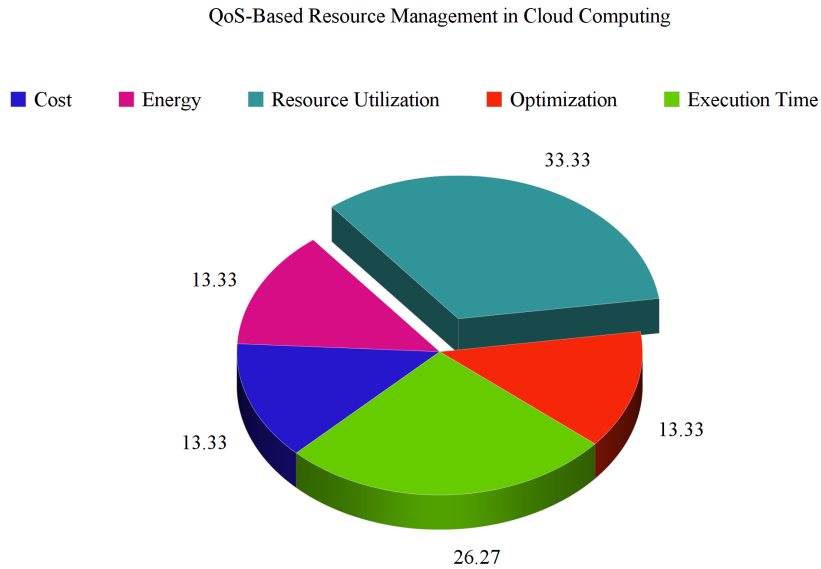


FIGURE 2.8: Usage rate of multi-objective functions under QoS-based RMS

Author Jala (Jala & Ramchand, 2019) strategy proposed in the task requires administrative viably resources with less strong depending on the accessibility of resources and oversee successful; survival depends on the effort and the high use of resources. The comparative study of QoS-based approach is shown in TABLE 2.8. The results of the QoS-based literature are shown in FIGURE 2.8. Where the resource utilization objective function usage is 40%, which is the maximum usage of in QoS-based literature in cloud resource management under cloud computing.

2.9 Autonomic Cloud Resource Management

This paper (Van et al., 2009) proposes an autonomous resource managers to control the virtual condition, which decouples the provision of resources from the dynamic arrangement of the virtual machine. Research point (Addis et al., 2010) is to manage resource allocation strategy for cloud virtualization condition ready to acknowledge the implementation and the off-exchange energy, providing certification prior accessibility of the cloud to the end-client. Author Xu (Xu et al., 2012) gifts brought together to support the learning approach, being a specific URL, for robotize virtual machine setup procedures and apparatus running in virtual machines. suitable methodology of continuous use automatic configuration clouds. This paper (Casalicchio et al., 2013) considers the problem where the cloud suppliers need to increase revenue, subject to a limit, the accessibility of the SLA, and VM movement imperative. The paper shows the heuristic settings, Near Optimal (nopt), with a NP-hard problem and check for side effects of exploration votes in correlation with the technique most suitable tasks (BF). Author Fargo (Fargo et al., 2014) presents the autonomous power and administration techniques execution framework for cloud considering the ultimate goal of a strong coordinating prerequisite applications with "just enough" source framework at runtime that a decrease in the strength of large, fast and also meet the nature of the administrative needs of cloud apps , This paper citejamshidi2014autonomic exploit fuzzy reasoning to empower subjective details of the flexibility rule for cloud-based programming. Likewise, this paper discusses the methodology controls hypothetical utilize type-2 fuzzy framework for reasons of flexibility under the vulnerability.

Author Sedaghat (Sedaghat et al., 2014) proposed (P2P) structure resource administration Peer-to-Peer, including a wide variety of specialists, overlaid as without scale set, in which the author Ssheikhalishahi (Sheikhalishahi et al., 2015) initially characterize and indicate a resource conflict metric for processing superior incredible burden as execution metrics in allocation algorithms and frameworks for most abnormal pile resource administration to address the fundamental problem in finding a skeleton. Author Viswanathan (Viswanathan et al., 2015) proposed job based on resource supply system provided with autonomous capabilities, in particular, self-association, self-improvement, and self-recover. Author Bruneo (Bruneo et al., 2015) proposed procedure, the right to assess the execution of measurement in the ground state as make sense as an instrument for the skeleton outline of IaaS clouds.

Author Elster (Elster et al., 2016) proposes administration clouds novel and design of conveyance in light of the standard self-association and self-administration of moving exertion send and enhancement of the shopper to stack the products that run on top of the cloud base, in which the author Jamshidi (Jamshidi et al., 2016) control-theory proposed methodologies relating to the implementation of cloud situation as a dynamic framework. In this paper (Singh et al., 2016), the effective energy autonomous cloud framework proposed for reservations energy productive

assets in the cloud server farm. The proposed work to think about energy as Quality of Service (QoS) parameters and consequently increase the ability of cloud assets by reducing the utilization of vitality.

TABLE 2.9: Differentiation of autonomic cloud RMS by evaluation parameters.

Frameworks	SVR	Cost	QoS	RU	Opt	Enrg	ET
Van et al. (2009)	✓	✓	×	×	×	×	×
(Addis et al., 2010)	×	×	×	×	×	✓	✓
(Xu et al., 2012)	✓	✓	×	×	×	×	×
(Casalicchio et al., 2013)	y	✓	✓	×	✓	×	×
(Fargo et al., 2014)	×	×	×	×	×	✓	×
(Jamshidi et al., 2014)	×	×	×	✓	×	×	×
(Sedaghat et al., 2014)	×	×	×	✓	×	×	✓
(Sheikhalishahi et al., 2015)	×	×	×	✓	×	×	✓
(Viswanathan et al., 2015)	×	✓	×	✓	×	×	×
(Bruneo et al., 2015)	×	✓	×	×	×	×	✓
(Elster et al., 2016)	×	×	×	✓	×	×	×
(Jamshidi et al., 2016)	×	✓	×	✓	×	×	×
(Singh et al., 2016)	×	✓	✓	×	✓	✓	×
(Ghobaei Arani et al., 2016)	✓	✓	×	×	×	×	×
(Tsfatsion et al., 2016)	×	×	×	×	×	✓	✓
(Gill et al., 2017)	✓	✓	×	×	✓	✓	✓
(Ghobaei Arani et al., 2018)	×	✓	×	✓	×	×	×
(Vashistha et al., 2018)	×	×	×	✓	×	×	×

Where SVR= SLA violation rate, QoS= Quality of Service, RU= Resource utilization, Opt= Optimization, Enrg= Energy, ET=Execution time, and y=Yes.

In this paper, (Ghobaei Arani et al., 2016) authors provide resources change the provision of the benefits of the cloud with the proposed sources of autonomous procurement approach that depends on the idea monitor control-Analyze-Plan-Execute (MAPE) loop, and they described the resource supply system for cloud environments.

Author Tsfatsion (Tsfatsion et al., 2016) online plan execution and the power estimator event that captures the intricate connection between the server application execution and control (separately), and the use of resources. Given this model, they designed two systems downsizing to decide the most advanced electrical design. Paper (Gill et al., 2017) offers self-design usability and resources, self recuperate by taking care of disappointment suddenly, the confidence to security attacks and self-downsizing to use the greatest resource, where the author (Ghobaei Arani et al., 2018) proposed approach Resource Provisioning for the benefits of hybrid cloud depends on the mix of ideas and Reinforcement Learning

autonomous register (RL). Likewise, they present a system for autonomous power source procurement, which is reused by the cloud layer model.

In (Vashistha et al., 2018) paper describes the utilization of self-adaptive methodology to streamline resource planning in the cloud. Moreover, this situation presents paper key which gives a view on the increasing difficulties in the administration of cloud resources. The comparative study of autonomic approaches is presented in TABLE 2.9. The results of the autonomic-based literature are shown in FIGURE 2.9. Where the cost objective function usage is 40%, which is the maximum usage of in autonomic-based literature in cloud resource management under cloud computing.

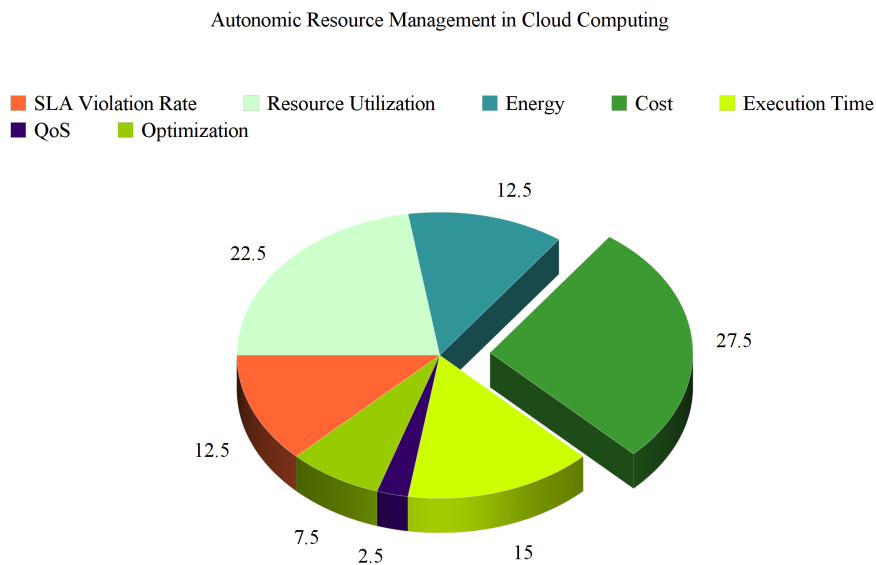


FIGURE 2.9: Usage rate of multi-objective functions under autonomic-based RMS

2.10 SLA-Based RMS

Author Wu (Wu et al., 2014) proposed client driven asset based SLA provision calculation for the cost limit to limit the resources and penalty costs and improve CSL by limiting SLA violations, in which the author Garg (Garg et al., 2014) address the issue of allotment of resources in the datacenter which runs sorts of peculiar use of the remaining load, especially non-smart applications and value-based. They propose a confirmation control and planning components, which reinforce the use of resources and benefits, and the guarantee that QoS to meet client needs as shown

in the SLA. Writer (Yali et al., 2015) propose confirmation control and resource planning calculations, which not only meet the QoS requirements of an application as determined in Service Level Agreements (SLAs), but also extend benefits for AAAS suppliers by offering financially savvy asset booking arrangements. This study (Kohne et al., 2016) examine the planning system VM-aware SLAs for cloud server farms. Interest levels are considered administrative resource utilization and accessibility.

The research objective (Mosa & Paton, 2016) is to build an advanced vitality and SLA-aware Virtual Machine (VM) methodology which allots more VMS situation for Physical Machines (PMS) in cloud server farms. These co-regulatory system utilization and progress of the vitality of Service Level Agreement (SLA) violations. Author Antonescu (Antonescu & Braun, 2016) present two calculation VM-new scale concentrating on Deis framework, which should ideally identify the condition scale most appropriate use the model application execution Convey get from assignment amazingly consistent in the benchmark hand, together with the imperative execution SLA determined, in which the author (Serrano et al., 2016) give specific permission dialect space to describe the SLA in cloud administration. They present the general methodology of control theory to supervise SLAs benefits of the cloud. They apply our methodology on MapReduce, locking, and internet business administration.

(Beloglazov & Buyya, 2016) Increase the use of physical resources and reduction of energy use in agriculture combines cloud server provides the majority of virtual machines in the data center cloud. Writer (Singh et al., 2017) SLA-Aware present-resource-autonomous administration strategy, called STAR, which mostly center on reducing the level of violations of the SLA to transport administration proficient clouds. (Cai et al., 2017) Propose a plan for the vitality of advanced booking SLA-aware, which allocates appropriately sized asset for MapReduce applications with design YARN. In this scheduling strategy, the authors consider the data information area to forgive MapReduce organize the movement, in which the author Panda (Panda & Jana, 2017) re-enact the proposed calculation utilizing benchmarks and datasets produced. The results clearly show that the proposed calculation legitimate balance between administrative costs and profits makespan in the examination with a different calculation. The comparative study of SLA-based approach is shown in TABLE 2.10.

TABLE 2.10: Differentiation of SLA-based RMS by evaluation parameters.

Frameworks	Cost	QoS	RU	Enrg	ET
(Wu et al., 2014)	✓	×	✓	×	×
(Garg et al., 2014)	×	✓	✓	×	×
(Yali et al., 2015)	✓	✓		×	×
(Kohne et al., 2016)	✓	×	✓	×	×
(Mosa & Paton, 2016)	×	×	✓	×	×
(Antonescu & Braun, 2016)	✓	×	✓	×	✓
(Serrano et al., 2016)	×	✓	×	×	×
(Beloglazov & Buyya, 2016)	×	×	✓	✓	×
(Singh et al., 2017)	×	✓	×	×	×
(Cai et al., 2017)	×	×	✓	✓	×
(Panda & Jana, 2017)	×	×	✓	×	✓

Where QoS= Quality of Service, RU= Resource utilization, Enrg= Energy, ET= Execution time, and y=Yes.

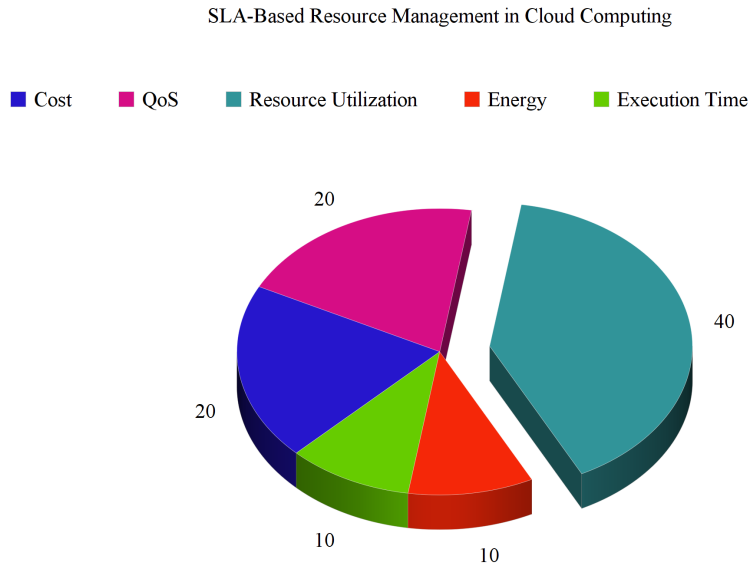


FIGURE 2.10: Distribution of various objective functions utilized in SLA-Based resource management in cloud computing

The results of the SLA-based literature are shown in FIGURE 2.10. Where the resource utilization objective function usage is 40%, which is the maximum usage of in SLA-based literature in cloud resource management under cloud computing.

2.11 Research Challenges

Challenges in resource management systems are categorized into the following:

1. **SLA Violation:** SLA means that it is an agreement between service providers and end users to meet QoS. While this QoS parameters does not meet the users requirement as per agreement, it known as SLA violation. SLA and QoS, these two parameters need to be improve to ensure the service-level guarantee (Wu et al., 2014).
2. **Scheduling time** need to be minimized so that the resources will be able to compute the task in less time.
3. **Resource utilization:** The use of resources is one of the difficult issues in the cloud, proper utilization of available resources the directly affect the organizations profit (Liu et al., 2010).
4. **Energy consumption rate** is very high due to large number of servers are used. It also affect the environment by releasing CO_2 (Arianyan et al., 2015).
5. **Fault-tolerant:** To avoid execution failure at run time, fault-tolerant mechanism should be improve and monitor as well, so that execution time can be minimize Egwutuoha et al. (2012).
6. **Resource cost** resource management is need to be developed as self-service and pay per use concepts are popular in current senaio (Sampaio & Barbosa, 2018) (Convolbo & Chou, 2016).
7. **Response Time:** While request submitted by the cloud user, the waiting time is high, due to this, users distraction is increasing.
8. **User trust:** User trust can be improve by ensuring to meet high QoS, low SLA violation rate, and more secured compute environment. item **Malicious Workload Identification:** The identification approach of duplicate request is required to avoid multiple execution of same workload (Garg et al., 2014).
9. **VM Migration:** When available virtual machine is not sufficient to compute the task submitted by end user, VM migration techniques come to the picture, through this, service provider can use VM's from other service provider to compute remaining task (L. X. Fang et al., 2014).
10. **Autonomic Management:** It manages resource through autonomic characteristics. Any one or combination of Some of characteristics can be used to improve the resource management technique (Singh & Chana, 2016).

Some of the above issues are targeted in this research and the following evaluation parameters are the outcomes of this study to evaluate the proposed research.

2.12 Evaluation Parameters

The outcome of the parameter based literature is presented in FIGURE 2.1. It presented that the distribution of parameters used to evaluate the research studied in this literature.

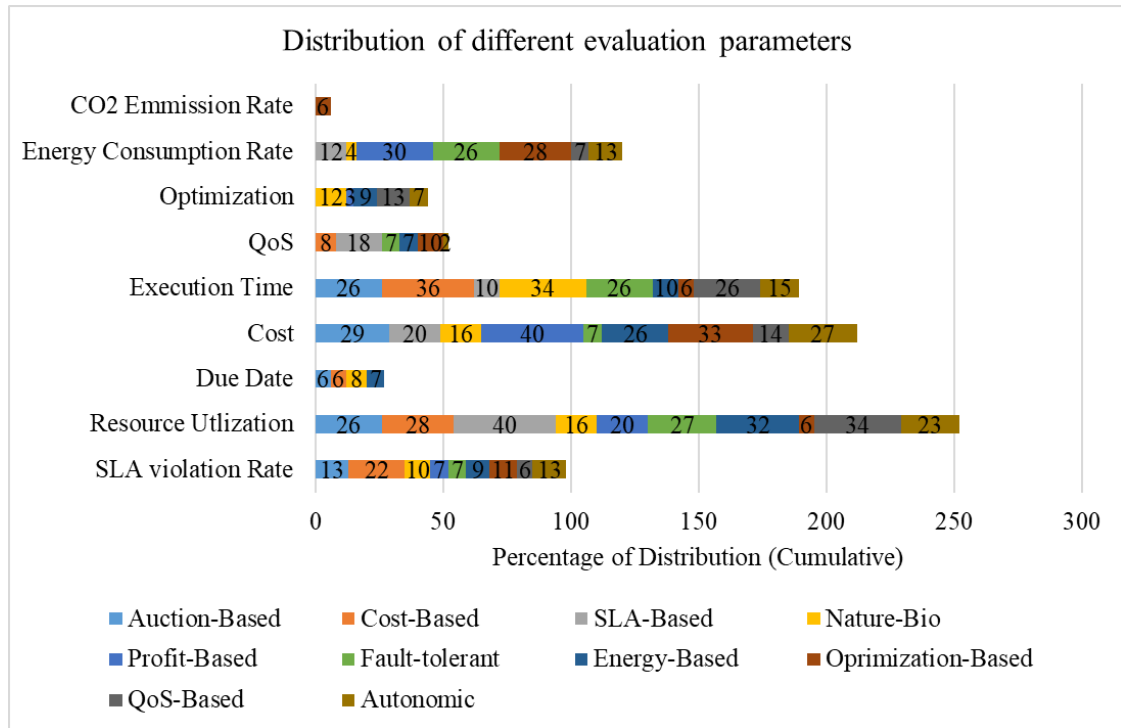


FIGURE 2.11: Evaluation Parameters Distribution cumulative

Based on the results of this study, the most popular parameters used to assess the effectiveness of resource management are: *rate of energy consumption, execution time, cost, SLA violation rate and use of resources.*

The least used evaluation parameters are presented in FIGURE 2.2 in percent wise. This analysis shows that QoS, CO₂, optimization, and due date are least used parameters.

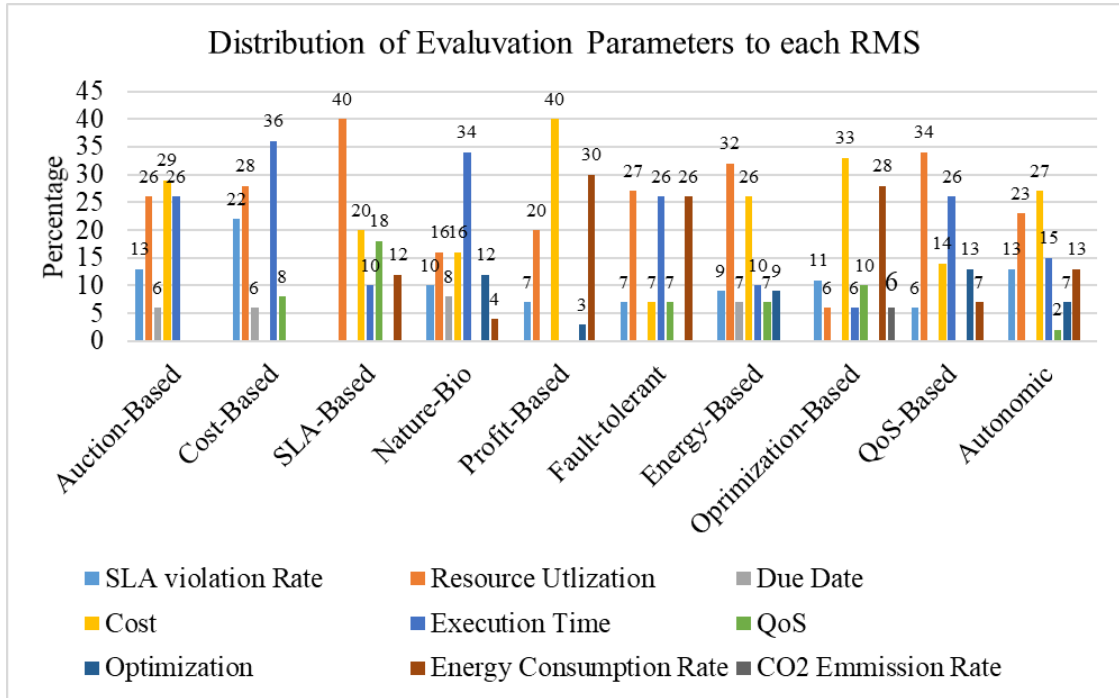


FIGURE 2.12: Evaluation Parameters Distribution for individual resource management techniques

The proposed research work SMART used the novel hybrid resource management technique based on modified antlion optimizer (Mirjalili, 2015) to minimize execution time, cost, energy consumption rate and maximize resource utilization.

2.13 Objective of the thesis

The above mentioned research challenges are the outcome of this extensive literature on resource management techniques in cloud computing. To make efficient RMS, some of the above challenges are considered in this research work. So, the objective of this thesis is to maximize the resource utilization while minimizing energy consumption rate, execution time, cost and SLA violation rate.

2.14 Summary

In this chapter, more than 250 articles from reputed journals carried out to have in depth knowledge of resource management frameworks in cloud computing. Near about 150 articles used to provide the study of this literature and results produced in tables and graph. In this study, we have identified evaluation parameters,

objective functions that should be achieved to improve the efficiency, which has been incorporated in next chapter.

Chapter 3

Methodology

Overview

Resource Management in cloud computing is one of the most essential feature to responsible for proper utilization of available resources to compute the workload request by the cloud users. This research introduces a new resource management method named SMART, which is further devided into Workload-filter, Self-optimization, Fault-tolerant, and Resource Management modules. In this chapter, Workload-filter module is discussed in details. In continue to this, the architecture of SMART and the simulation/implementation enviroment is presented with system configuration.

3.1 The Proposed Research Methodology

The flow of proposed research work is presented in FIGURE 3.1. SMART is having the following modules:

1. Workload-Filter,
2. Self-optimization,
3. Fault-tolerant,
4. Resource Management, and
5. Performance Analysis.

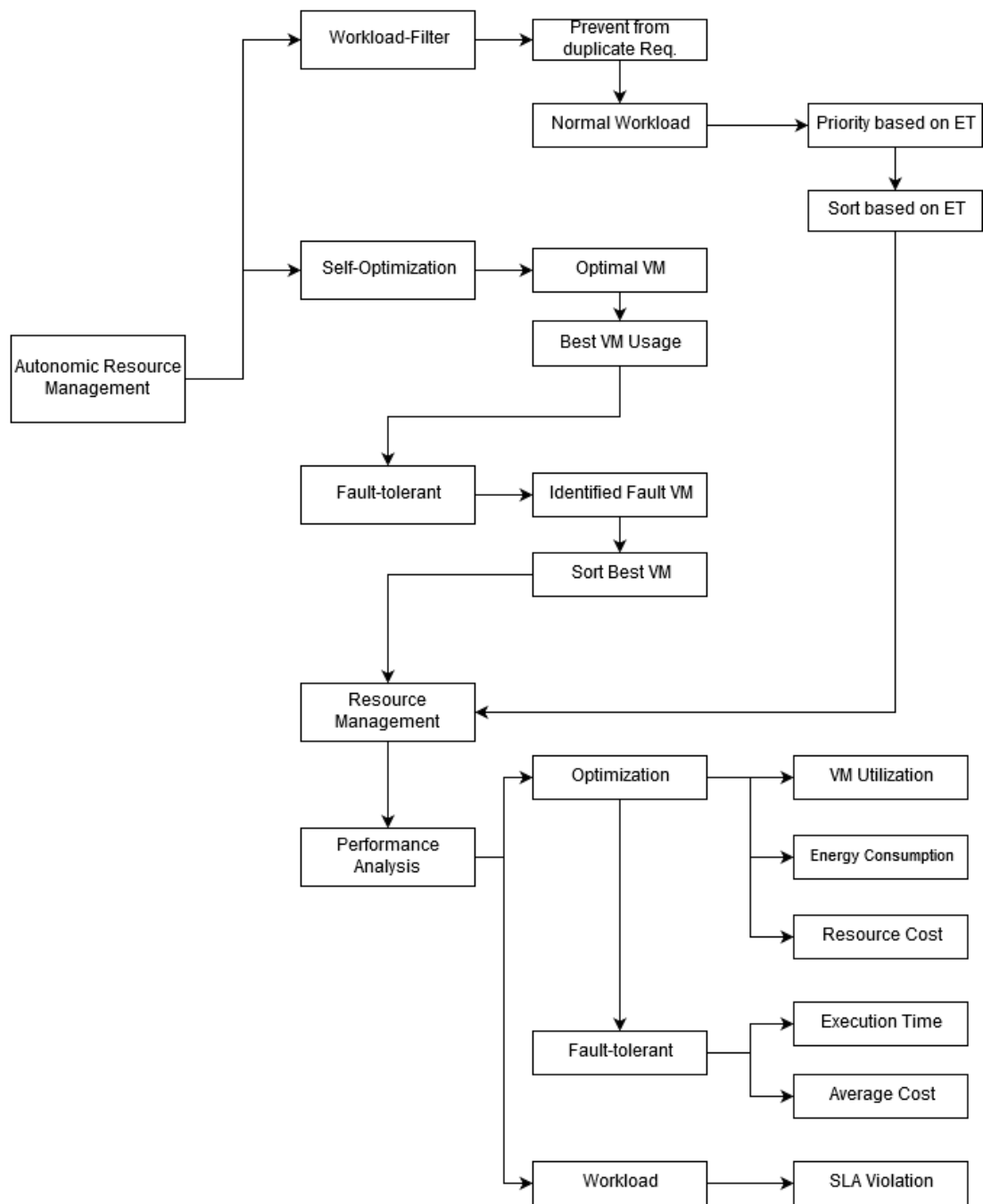


FIGURE 3.1: SMART Methodology

3.2 Workload-Filter

This module basically identifies the duplicate workload request, and separate it from the container. Then the workload execution priority of each workload is considered and sort the workloads according to the priority based on execution time and send to the resource management module for scheduling.

3.2.1 Workload Dataset

The workload, which is to be submit to the cloud for resource request is created and presenting to the following TABLE 3.1.

TABLE 3.1: Dataset ([Singh et al., 2016](#)) ([Gill et al., 2017](#))

Cloud Workload	Quality of Service
Web Services	Reliable Storage, High Network Bandwidth, High Availability, Technological Computing
E-Com	Internet Accessibility, Reliability
Software Project Development	High Availability, High Network Bandwidth, Visibility
Graphics Oriented	Persistence, Serviceability
Endeavor Software	Data Backup, Testing Time
Storage And Backup Services	Visibility, Integrity, Testing Time
Critical Internet Applications	Customizability, Reliability
Productivity Applications	Testing Time, Integrity, Customer Confidence Level
Performance	Testing Time, Cost, Energy, Resource Utilization And SLA Violation
Online Transaction	Processing, Security, High Availability, Internet Accessibility, Usability
Central Finance Service	Security, High Availability, Changeability, Usability
Mobile Computing Services	High Availability, Reliability

3.2.2 Algorithm

The following algorithm is degned for finding the redundant workload.

Algorithm 1 Workload Filtration**Input:** worloads_list**Output:** Filtered_workloads (No duplicate workload)**Start**

```

  /* Initialize Workloads                                     */
W ← W1, W2, W3,.....Wn while i < n do
  | if (Wfrequency = 1) then
  |   | assign to Filtered_workloads;
  | end
  | else
  |   | identify as duplicate worklod and
  |   | seperate from workload_list;
  | end
end
Return Filtered_workloads
End

```

Here n = Workload size (Number of Workloads)

W₁ means workload with id = 1.

W_{frequency} is frequency of workload.

3.2.3 User-Priority based on Execution Time

User priority is based on excetion time tken in this research. The execution time for each workload is obtain through following equation which is used in Chopper (Gill et al., 2017):

$$ET = \sum_{i=1}^n \frac{RC_{(i)} - RS_{(i)}}{n} \quad (3.1)$$

Where RC_(i) is request for workload completion time and RS_(i) is the workload submission time, and n is number of workloads.

3.3 Self-optimization

This module is responsible to find the optimal resource from the resource pool. In this research, in the case of CloudSim 3.0, resource means virtual machine VM and in the case of amazon web service AWS, resource means EC2 instances. This module is working on Antlion Optimizer (Mirjalili, 2015), and produce the optimal resource for scheduling. The detailed description of self-optimization technique is discussed in chapter 4.

3.4 Fault-tolerant

This module identifies the fault resource based on the fitness value of each resource. If the resource is capable enough to execute the workloads submitted by cloud users, then only it will be allowed to process the resources for scheduling. This module is responsible to avoid the execution failure which improves the system performance and reduces the execution time. The detailed description of fault-tolerant technique is discussed in chapter 4.

3.5 Resource Management

This module is responsible for workload scheduling with optimal resources. It takes inputs from Workload-Filter and Fault-tolerant module and executes the scheduling algorithm. The detailed description of Resource Management technique is discussed in chapter 5.

3.6 Performance Analysis

To evaluate the performance of proposed research work SMART, the following performance analysis parameters are taken into consideration which is also mentioned in Introduction section 1.10.

1. Resource Utilization,
2. Execution Time,
3. Energy Consumption,
4. SLA Violation rate,
5. Average Cost,

The performance analysis is conducted for proposed work SMART, SOCCER, and CHOPPER, which is presented in chapter 5.

3.7 SMART-Architecture

SMART aims to provide efficient autonomic resource management through workload filter (protection), self-optimization, and fault-tolerant. The proposed research is simulated SMART algorithms in CloudSim 3.0 and tested in AWS EC2

Instance.

The objective of SMART is

“To attain the better system by incorporating an efficient and intelligent resource scheduling algorithm in cloud computing”.

The architecture of the proposed research work SMART is presented in FIGURE 3.2.

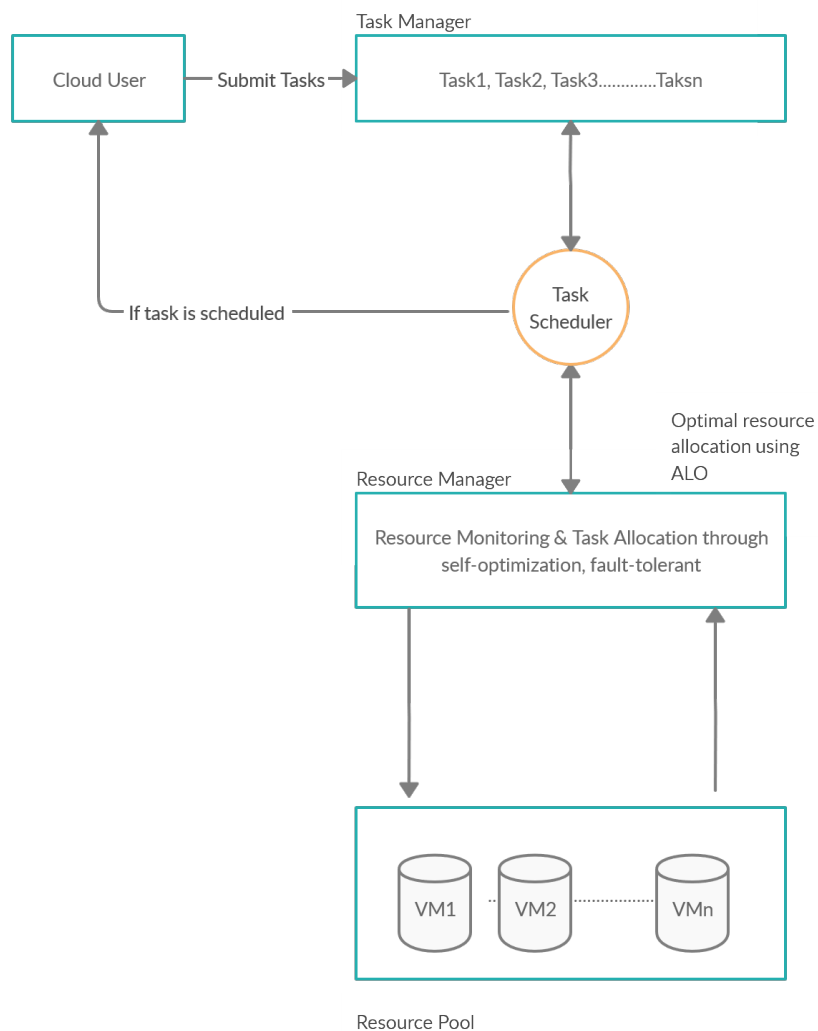


FIGURE 3.2: Architecture of proposed research work SMART

3.8 Simulation & Test Environment

The proposed research work is simulated and tested in CloudSim 3.0, and also validate in AWS EC2 environment. The configuration of both the environments are discussed in next sub section.

3.8.1 Simulation Environments

In R&D, it is not always possible to have the actual cloud infrastructure for performing experiments. For any research scholar, academician or scientist, it is not feasible to hire cloud services every time and then execute their algorithms or implementations. For the purpose of research, development and testing, open source libraries are available, which give the feel of cloud services. Nowadays, in the research market, cloud simulators are widely used by research scholars and practitioners, without the need to pay any amount to a cloud service provider. Many open source cloud simulators are available, we simulates our proposed work in CloudSim 3.0 ([Calheiros et al., 2011](#)) toolkit.

3.8.2 Cloudsim Simulation Toolkit

CloudSim is a famous simulator for cloud parameters developed in the CLOUDS Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne. The Cloudsim library is used for the following operations:

1. Large scale cloud computing at data centres
2. Virtualised server hosts with customisable policies
3. Support for modelling and simulation of large scale cloud computing data centres
4. Support for modelling and simulation of virtualised server hosts, with customisable policies for provisioning host resources to VMs
5. Support for modelling and simulation of energy-aware computational resources
6. Support for modelling and simulation of data centre network topologies and message-passing applications
7. Support for modelling and simulation of federated clouds

8. Support for dynamic insertion of simulation elements, as well as stopping and resuming simulation
9. Support for user-defined policies to allot hosts to VMs, and policies for allotting host resources to VMs
10. User-defined policies for allocation of hosts to virtual machines

The major limitation of CloudSim3.0 is the lack of a graphical user interface (GUI). But despite this, CloudSim is still used in universities and the industry for the simulation of cloud-based algorithms.

System Configuration

The proposed work is simulated in CloudSim 3.0. The system configuration in which the simulator installed is presenting in TABLE 3.2.

TABLE 3.2: Physical system configuration in which simulator installed.

Name of PM	Configuration	Specifications	System Type	Operating System
HP 240 G5	Intel(R) Core(TM) i3 2.0 GHz	4.00 GB RAM 1 TB HDD	64 bit OS	Windows 10

The configuration of the simulation background is presenting in TABLE 3.3.

TABLE 3.3: Simulation environment details.

Number of PM	Number of VM	Number of Host
10	20	10

The initialization of maximum limit of CPU, RAM, and Bandwidth for PM and VM in simulator are presenting in TABLE 3.4.

TABLE 3.4: Configuration of simulation environment for CPU, RAM, and Bandwidth for PM_{\max} and VM_{\max} .

Machine	CPU	RAM	Bandwidth	Cost
PM_{\max}	1000 mips	16384 mbps	10000 bps	\$600
VM_{\max}	50 mips	512 mbps	1000 bps	\$30

3.8.3 Amazon Web Services Environment

Amazon Web Services (AWS) is the market leader in IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) for cloud ecosystems, which can be combined to create a scalable cloud application without worrying about delays related to infrastructure provisioning (compute, storage, and network) and management.

With AWS user can select the specific solutions as per need, and only pay for exactly what they use, resulting in lower capital expenditure and faster time to value without sacrificing application performance or user experience. Amazon offers many services for application development and analytics. Here are some key building blocks in the AWS environment against user needs are EC2: Server configuration and hosting, Amazon S3: Data storage and movement, Elastic Load Balancing, Elastic Block Store (EBS) etc.

The proposed research SMART is deployed in amazon web services AWS with compute service elastic cloud computing EC2 instance. The description of EC2 instance is presented in TABLE 3.5:

3.9 Summary

In this chapter, the different module of SMART is discussed in brief. The research methodology is explained and the SMART-architecture is presented. The proposed research work SMART is simulated in CloudSim 3.0 and also validated in a real-time cloud environment in AWS EC2 and discussed in this chapter. The next chapter is all about how the self-optimization technique is modeled with modified Antlion optimizer and how the fault-tolerant technique is applied in SMART.

TABLE 3.5: Description EC2 Instance

Instance ID	i-0a067eaf356b357b7 (ALO_scheduler)
Instance state	running
Instance type	t2.micro
Elastic IPs	1
Availability zone	us-east-1c
Security groups	launch-wizard-3. view inbound rules. view outbound rules.
Scheduled events	No scheduled events
AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200112(ami-07ebfd5b3428b6f4d)
Platform	Linux/UNIX
IAM role	-
Key pair name	scheduler
Owner	564005883562
Launch time	Jan 15, 2020 at 2:28:24 PM UTC+5:30 (575 hours)
Termination protection	False
Lifecycle	normal
Monitoring	basic
Alarm status	None
Kernel ID	-
RAM disk ID	-
Placement group	-
Partition number	-
Virtualization	hvm
Reservation	r-02b5a3693f5a44f91
AMI launch index	0
Tenancy	default
Host ID	-
State transition reason	-
State transition reason message	-
Stop - Hibernation behavior	Disabled
Number of vCPUs	-
Public DNS (IPv4)	ec2-172-31-83-149.compute-1.amazonaws.com
IPv4 Public IP	172.31.83.149
IPv6 IPs	-
Private DNS	ip-172-31-83-149.ec2.internal
Private IPs	172.31.83.149
Secondary private IPs	-
VPC ID	vpc-1313f378
Subnet ID	subnet-0380294f
Network interfaces	eth0
Source/dest. check	True
T2/T3 Unlimited	Disabled
EBS-optimized	False
Root device type	ebs
Root device	/dev/sda1
Block devices	/dev/sda1
Capacity Reservation Settings	Open

Chapter 4

Self-optimization and Fault-tolerant Mechanism

Overview

This chapter is described the modules of SMART, which are Self-optimization, and Fault-tolerant. The goal of self-optimization is to maximize the resource utilization and minimize the energy consumption and cost. Minimizing the energy consumption has a momentous outcome on the total productivity, reliability and availability of the system. Therefore, minimizing this energy consumption does not only reduce the huge cost and improves system reliability, but also helps in protecting the natural environment. The cloud workloads sent by cloud users are executed only the above constraints. Self-optimization allocate resources optimally. In continue to this, Fault-tolerant mechanism is responsible to identify the duplicate workload request by cloud user, to manage workloads in an efficient way.

4.1 Self-optimization

The objective of this module is

To maximize the resource utilization through self-optimization.

4.1.1 Formal Optimization Model

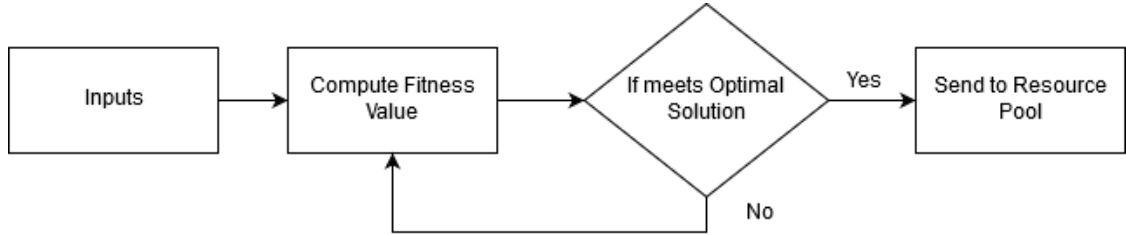


FIGURE 4.1: Formal Optimization Model

The flow of formal optimization model is representing through FIGURE 4.1. In general, to obtain the optimal solution, population and fitness values has to be calculated. On the basis of these values if the method meets the optimization criteria then it will stop to search the solution and resources will be sent to the pool else the process need to be regenerate the population and calculate the fitness values. In this work, we proposed modified Antlion Optimizer (Mirjalili, 2015).

4.1.2 Modified ALO based Self-optimization

In this research work, we used antlion optimizer (ALO)(Mirjalili, 2015) for the fittest VM selection. According to this, The antlion algorithm mimics interaction between antlions and ants in the trap. To model such interactions, ants are required to move over the search space, and antlions are allowed to hunt them and become fitter using traps. Since ants move stochastically in nature when searching for food, a random walk is chosen for modelling ants' movement.

During optimization, the following conditions are applied:

1. Ants (VM) move around the search space using different random walks.
2. Random walks are applied to all the dimension of ants (VM).
3. Random walks are affected by the traps of antlions (Best VM).
4. Antlions (Best VM) can build pits proportional to their fitness (the higher fitness, the larger pit).
5. Antlions (Best VM) with larger pits have the higher probability to catch ants (VM).
6. Each ant (VM) can be caught by an antlion (Best VM) in each iteration and the elite (fittest antlion).

7. The range of random walk is decreased adaptively to simulate sliding ants towards antlions.
8. If an ant becomes fitter than an antlion, this means that it is caught and pulled under the sand by the antlion.
9. An antlion repositions itself to the latest caught prey and builds a pit to improve its change of catching another prey after each hunt

4.1.3 Modified ALO Operators

In this research for best VM selection we used antlion optimizer (ALO) (Mirjalili, 2015). After the centroid positions are finalized in the clustering process, consider the nodes which are at the nearest distance and also the next nearest distance from the centroid by using ALO optimization. In this method, the initialization is done by using randomly selection of antlions and ants (VM). After that the multi objectives (resource utilization, energy and cost) is calculated between each VMs, from that the fitness value of ant and antlion is calculated and antlion fitness value sorted to find the best antlion is named as *elite* antlion (best VM). The following optimization equation is used to store the position of ants:

$$M_{\text{Ant}} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1d} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & A_{n3} & \dots & A_{nd} \end{bmatrix} \quad (4.1)$$

Where M_{Ant} is the position of each ant store in the matrix. $A_{i,j}$ shows the value of the j^{th} variable (dimension) of i^{th} ant, n is the number of ants, and d is the number of variables. For evaluating each ant, a fitness (objective) function is utilized during optimization and the following matrix stores the fitness value of all ants:

$$M_{\text{OA}} = \begin{bmatrix} f([A_{11} & A_{12} & A_{13} & \dots & A_{1d}]) \\ f([A_{21} & A_{22} & A_{23} & \dots & A_{2d}]) \\ \dots & \dots & \dots & \dots & \dots \\ f([A_{n1} & A_{n2} & A_{n3} & \dots & A_{nd}]) \end{bmatrix} \quad (4.2)$$

where M_{OA} is the matrix for saving the fitness of each ant, $A_{i,j}$ shows the value of j^{th} dimension of i^{th} ant, n is the number of ants, and f is the objective function. In addition to ants, we assume the antlions are also hiding somewhere in the search space. In order save their positions and fitness values, the following matrices are utilized:

$$M_{\text{Antlion}} = \begin{bmatrix} AL_{11} & AL_{12} & A_{13} & \dots & AL_{1d} \\ AL_{21} & AL_{22} & A_{23} & \dots & AL_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ AL_{n1} & AL_{n2} & A_{n3} & \dots & AL_{nd} \end{bmatrix} \quad (4.3)$$

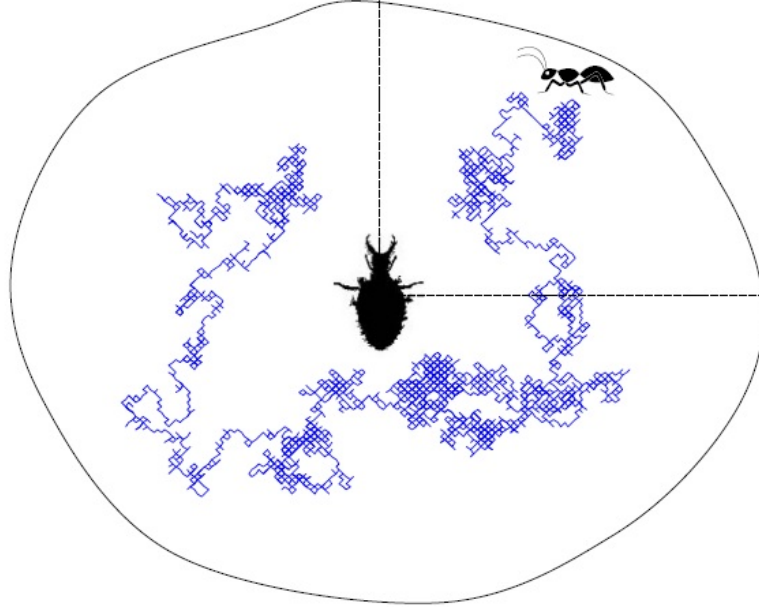


FIGURE 4.2: Roulette wheel method (Mirjalili, 2015)

where a_i is the minimum of random walk of i^{th} variable, d_i is the maximum of random walk in i^{th} variable, c_i^t is the minimum of i^{th} variable at t^{th} iteration, and d_i^t indicates the maximum of i -th variable at t -th iteration. Equ. 4.7 should be applied in each iteration to guarantee the occurrence of random walks inside the search space.

4.1.5 Building Trap (Fitness Value)

Consider that the ants are trapped in any one of the randomly selected antlion. The roulette wheel (as shown in FIGURE 4.2) is used to select a best antlion from its population based on fitness calculation. This highly fit antlion to catch the ants with high probability. The resource utilization is found out by using the below equations:

CPU Utilization

The CPU utilization of any VM can be calculated by CPU value of current VM divided by maximum allocated CPU value for VM. The CPU utilization by each VM is calculated by Equ. 4.8.

$$U_{VMi}^{CPU} = \sum_{i=1}^m \frac{VM_i^{CPU}}{VM_{max}^{CPU}} \quad (4.8)$$

Where U_{VMi}^{CPU} is CPU utilization by i^{th} VM, VM_i^{CPU} is initial CPU value of i^{th} VM, $VM_{max}^{CPU} = 50 \text{ mips}$, is maximum CPU that any VM can utilize (as stated in **TABLE 3.3**),

The range of CPU utilization will be $0 < U_{VMi}^{CPU} < 1$.

RAM Utilization

The RAM utilization of any VM can be calculate by RAM value of VM divide by maximum allocated RAM utilization for VM. The RAM utilization by each VM is calculation by Equ. 4.9.

$$U_{VMi}^{RAM} = \sum_{i=1}^m \frac{VM_i^{RAM}}{VM_{max}^{RAM}} \quad (4.9)$$

Where U_{VMi}^{RAM} is RAM utilization by i^{th} VM, VM_i^{RAM} is initial RAM value of i^{th} VM, $VM_{max}^{RAM} = 512 \text{ mb}$, is maximum RAM that any VM can utilize (as stated in **TABLE 3.3**),

The range of RAM utilization will be $0 < U_{VMi}^{RAM} < 1$.

Bandwidth Utilization

The bandwidth(BW) utilization of any VM can be calculate by bandwidth value of VM divide by maximum allocated bandwidth for VM. The bandwidth utilization by each VM is calculation by Equ. 4.10.

$$U_{VMi}^{BW} = \sum_{i=1}^m \frac{VM_i^{BW}}{VM_{max}^{BW}} \quad (4.10)$$

Where U_{VMi}^{BW} is Bandwidth utilization by i^{th} VM, VM_i^{BW} is initial Bandwidth value of i^{th} VM, $VM_{max}^{BW} = 1000 \text{ kbps}$, is maximum Bandwidth that any VM can utilize (as stated in **TABLE 3.3**),

The range of Bandwidth utilization will be $0 < U_{VMi}^{BW} < 1$.

VM Utilization

The fitness of each VM is given by resource utilization, and it is mean of CPU, RAM, and Bandwidth utilization of each VM, and it is find out by using the below condition:

$$U_{VMi} = \sum_{i=1}^m \frac{U_{VMi}^{CPU} + U_{VMi}^{RAM} + U_{VMi}^{BW}}{3} \quad (4.11)$$

Where U_{VMi} is VM by i^{th} VM, VM_{VMi}^{CPU} is CPU utilization value by i^{th} VM, VM_{VMi}^{RAM} is RAM utilization value by i^{th} VM, VM_{VMi}^{BW} is Bandwidth utilization value by i^{th} VM,

The range of VM utilization will be $0 < U_{VMi} < 1$.

Therefore the first objective function of is :

Maximize: $f(U_{VMi})$

Constraint: $0 \leq \alpha \leq 50, 0 \leq \beta \leq 512, 0 \leq \gamma \leq 1000$.

Energy Consumption

The energy consumption rate is obtain through Equ. 4.12, which is used in Soccer (Singh et al., 2016).

$$EnC = (EnC_{\max} - EnC_{\min}) * U_{VMi} + EnC_{\min} \quad (4.12)$$

Where EnC is energy consumption rate, EnC_{\max} is the energy consumption at the peak load (or 100% utilization), and EnC_{\min} is the minimum energy consumption in the active/idle mode (or as low as 1% utilization). In this research, EnC_{\max} is 1 kWh and EnC_{\min} is $0 < EnC_{\min} < 0.05$.

Therefore the Second objective function of is :

Minimize: $f(EnC)$ **Constraint:** $0 < EnC \leq 100\%$.

Resource Cost

The Resource cost is obtain through Equ. 4.13, which is used in Soccer (Singh et al., 2016).

$$RC = c * EnC \quad (4.13)$$

Where c is initial cost that is \$30, as mentioned in **TABLE 3.3**, RC is Resource Cost and EnC is Energy Consumption Rate.

Therefore the third objective function is :

Minimize: $f(RC)$

4.1.6 Entrapment of ant in trap

Antlions are building their trap with their fitness value. If the ant is inside the trap then it shoots the sand outside the center of the pit. It makes the ant slide down when it tries to escape from the antlion.

4.1.7 Catching preys and rebuilding trap

Ant reaches the bottom of the pit, and then the antlion pulls it in sand and eats its body. Next the antlion modify its position and build a new trap to catch a new prey. Antlion catches ant, when the fitness of ant is greater than the fitness of antlion. It's given by,

$$AL_j^t = A_i^t \quad (4.14)$$

Constraint *if* $fitness(A_i^t) > fitness(AL_j^t)$

4.1.8 Algorithm

Algorithm 2 Self-optimization

Input: list of Ant's (workloads), list of Antlion's (VM's).

Output: Elite_Antlion (the best solution for tasks allocation on VM's)

Start

```

    /* Antlion's (VM's) Initialization */
    VM ← VM1, VM2, VM3.....VMk; // k is number of Antlion's (VM's)
    Compute Fitness of Ant's & Antlion's
    /* CPU, RAM, & Bandwidth of each VM */
    for i=1 to k do
        | compute CPU usage of VMi using Equ. (4.8)
        | compute RAM usage of VMi using Equ. (4.9)
        | compute Bandwidth usage of VMi using Equ. (4.10)
    end
    /* Antlion optimizer */
    Find the fittest Antlion's (VM's) & assume it Elit_Antlion's; // Optimum
    Antlion's (VM's)
    while end criterion does not meet do
        | for every Ant's do
            | Select an Antlion using roulette wheel
            | Update the position of Ant using Equ. (4.15)
        end
        Compute the fitness of all Antlion's; // CPU, RAM, & Bandwidth of each
        VM
        Replace an Antlion with its corresponding Ant if it become fitter using Equ.
        (4.14), and assign it to Elite_Antlion
        Elite_Antlion = Antlion
    end
    Return Elite_Antlion
    End

```

Here the fittest antlion is named as elite antlion. After this ants movements are remodelled by

$$A_i^t = \frac{R_A^t + R_E^t}{2} \quad (4.15)$$

Where R_A^t and R_E^t are the random walk of ant lion and elite antlion at 't' iteration is used to update the position of ant.

4.2 Fault-tolerant Mechanism

Fault-tolerance innovation is a capacity of a computer framework, electronic framework or system to convey continuous service, despite of its components failing. Adaptation to non-critical failure likewise settle potential service interruptions identified with programming or logical errors. The reason is to prevent the system from a single point of failure. Fault-tolerance frameworks are intended to make up for various failures. Such frameworks naturally distinguish a failure of the computer processor unit, I/O subsystem, memory cards, motherboard, control supply or system parts. The failure point is recognized, and a backup segment or strategy promptly has to take place with no loss of service.

The objective of this module is

*To design a resource management system RMS in cloud which improves performance in terms of execution time and average cost through fault tolerance by reducing or avoiding the impact of failures on execution.*¹

4.2.1 Fault Detection Procedure

The flow of proposed fault detection procedure is presented in FIGURE 4.3.

¹An Automated Self-Healing Cloud Computing Framework for Resource Scheduling. *International Journal of Grid and High Performance Computing*. Vol (13), Issue (2), pp: 47-64. Indexed by SCOPUS, ESCI, IET Inspec. Publisher: IGI Global. USA.

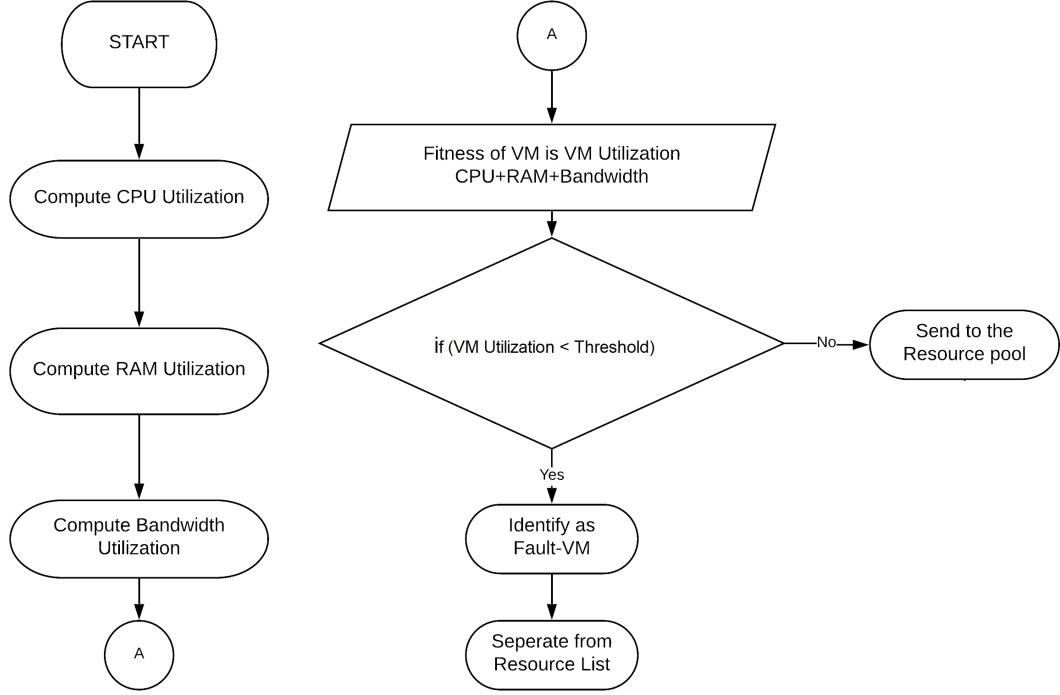


FIGURE 4.3: Fault VM computation.

4.2.2 Threshold

The threshold value for identification of VM's, which have minimum utilization value that can not be consider for allocation. In this research, we have set threshold value of CPU, RAM, and Bandwidth utilization individually.

Threshold value for Very low utilization capacity of CPU, RAM, and Bandwidth utilization is 10%. The threshold for CPU, RAM, and Bandwidth is obtained through following euations:

Threshold for CPU

$$threshold_{CPU} = VM_{max}^{CPU} * \delta \quad (4.16)$$

Where $threshold_{CPU}$ is threshold for CPU utilization, VM_{max}^{CPU} is maximum CPU utilization, and δ is constant multiplier, and taken 10% in this research, and $VM_{max}^{CPU} = 50$ mips.

Threshold for RAM

$$threshold_{RAM} = VM_{max}^{RAM} * \delta \quad (4.17)$$

Where $threshold_{RAM}$ is threshold for RAM utilization, VM_{max}^{RAM} is maximum RAM utilization, and δ is constant multiplier, and taken 10% in this research, and $VM_{max}^{RAM} = 512$ mb.

Threshold for Bandwidth

$$threshold_{BW} = VM_{max}^{BW} * \delta \quad (4.18)$$

Where $threshold_{BW}$ is threshold for Bandwidth utilization, VM_{max}^{BW} is maximum BW utilization, and δ is constant multiplier, and taken 10% in this research, and $VM_{max}^{BW} = 1000$ kbps.

4.2.3 Algorithm

Algorithm 3 Fault-tolerant

Input: list of Ant's (workloads), list of Antlion's (VM's).

Output: Fault_Antlion, Best_Antlion_list (the best solution for tasks allocation on VM's)

Start

```

    /* Antlion optimizer */
Initialize Antlion's (VM's)
    VM ← VM1, VM2, VM3.....VMk

/* Calculate Fitness of each Antlion's (VM's) */
for i = 1 to k do
    compute CPU usage of VMi using Equ. (4.8)
    compute RAM usage of VMi using Equ. (4.9)
    compute Bandwidth usage of VMi using Equ. (4.10)
end
/* Compute Fault_Antlion */
for i = 1 to k do
    if UVMiCPU < thresholdCPU && UVMiRAM < thresholdRAM && UVMiBW < thresholdBW
        then
            add VMi to Best_Antlion_list
        end
    else
        assign to Fault_Antlion and remove from Antlion_list
    end
end
end
Return Fault_Antlion
Return Best_Antlion_list
End

```

4.3 Summary

In this chapter, the Antlion optimizer is explained which is modified for cloud resource management. The multiobjective functions which need to be optimized and self-optimization algorithm is discussed to find the optimal resource. In continue to this, Fault-tolerant method is also described in this chapter with the threshold criteria for each VM parameters (CPU, RAM, and Bandwidth). The resource management technique is discussed in next chapter.

Chapter 5

Resource Management

Overview

Autonomic resource management is a capacity to enhance usage of resources (VM's for this research) and client fulfillment in autonomic frameworks. Based on state of art survey, which discussed in chapter 2, it is found that the resource management in cloud is an essential requirement for service provider and cloud user as well. The resource management in cloud need to be low in cost and execution. The proposed research work SMART is simulated in Clousim 3.0 and implemented in AWS EC2 instances. The experimental results are compared based on the following performance metrics:

1. SLA violation rate,
2. Execution time,
3. Resource utilization,
4. Energy consumption,
5. Cost

The objective of this chapter is to *“Attain the better system by incorporating an efficient and intelligent resource scheduling algorithm in cloud computing”*

5.1 Scheduling

The workloads submitted by the users are processed through SMART and duplicate workloads identified through workload-filter and separated from the task manager.

The initialized VM's are processed with self-optimization and fault VM's identified, and separated from the resource pool. The Elite VM's and workloads are scheduled through round-robin scheduling algorithm.

5.2 Algorithm

The following algorithm is designed for autonomic resource management

5.2.1 Time Complexity

The time complexity is the computational complexity that describes the amount of time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of elementary operations performed by the algorithm are taken to differ by at most a constant factor. The complexity class QP consists of all problems that have quasi-polynomial time algorithms. It can be defined in terms of TIME as follows:

$$QP = \bigcup_T \text{Time}(2^{\log n}) \quad (5.1)$$

5.3 Results and Analysis

SMART simulates in cloudsim 3.0 toolkit as well as simulation is tested in Amazon Web Services AWS. The system obtained optimal and fault VM based on modified Ant Lion Optimizer, and identified redundant workloads through workload-filter. SMART's objective is to *implement a resource management system to improve the fault tolerance mechanism, self-adaptability and maximize resource utilization in cloud computing*. This section is organized based on performance metrics.

5.3.1 Performance Metrics

To observe the efficiency of SMART system based on the following metrics:

1. Resource Utilization by workloads
2. Execution Time

Algorithm 4 Self-managment Aware Autonomic Resource Management**Input:** list of Ant's (workloads), list of Antlion's (VM's).**Output:** the best solution for tasks allocation on VM's**Start**

```

    // Call Workload Filtraion
/* Initialize Workloads */
 $W \leftarrow W_1, W_2, W_3, \dots, W_n$  while  $i < n$  do
    if ( $W_{frequency} = 1$ ) then
        | assign to Filtered_workloads;
    end
    else
        | identify as duplicate worklod and
        | seperate from workload_list;
    end
end
/* Call Fault-tolerant */
Initialize Antlion's (VM's)
 $VM \leftarrow VM_1, VM_2, VM_3, \dots, VM_k$ 

/* Calculate Fitness of each Antlion's (VM's) */
for  $i = 1$  to  $k$  do
    | compute CPU usage of  $VM_i$  using Equ. (4.8)
    | compute RAM usage of  $VM_i$  using Equ. (4.9)
    | compute Bandwidth usage of  $VM_i$  using Equ. (4.10)
end
/* Compute Fault_Antlion */
for  $i = 1$  to  $k$  do
    | if  $U_{VM_i}^{CPU} < threshold_{CPU}$  &&  $U_{VM_i}^{RAM} < threshold_{RAM}$  &&  $U_{VM_i}^{BW} < threshold_{BW}$ 
    | then
    | | add  $VM_i$  to Best_Antlion_list
    | end
    | else
    | | assign to Fault_Antlion and remove from Antlion_list
    | end
end
/* Call Self-optimization */
Compute Fitness of Ant's & Antlion's // CPU, RAM, & Bandwidth of each VM
Find the fittest Antlion's (VM's) & assume it Elit_Antlion's ; // Optimum
while end criterion does not meet do
    | for every Ant's do
    | | Select an Antlion using roulette wheel
    | | Update the position of Ant using Equ. (4.15)
    | end
    | Compute the fitness of all Antlion's ; // CPU, RAM, & Bandwidth of each VM
    | Replace an Antlion with its corresponding Ant if it become fitter using Equ. (4.14), and assign it to Elite_Antlion
    | Elite_Antlion = Antlion
end
End

```

3. SLA Violation Rate
4. Energy Consumption Rate
5. Cost

5.3.2 Validation of SMART

The proposed system SMART is validated for Execution Time, Resource Utilization, SLA Violation Rate, Cost, and Energy Consumption Rate and used for validation of performance of SMART. The validation performed the different number of experiments in different type of parameters by comparing SMART with existing autonomic resource management techniques. Experiment has been conducted with different number of workloads (10–4000) for validation of different performance metrics has been evaluated. For each performance metrics, minimum 30 tests has been performed. The following existing resource management approaches have been considered to validate SMART:

5.3.2.1 SOCCER

Self-Optimization of Energy-efficient Cloud Resources (SOCCER) ([Singh et al., 2016](#)) is an autonomic resource management technique which schedules the resources automatically by optimizing energy consumption. Scheduling rules have been designed using the concept of fuzzy logic to calculate the priority of workload execution. Large number of rules is generated for every request, so it is very difficult to take an effective decision in timely manner. SOCCER always executes the workloads with highest priority based on energy consumption, in which workloads with lowest priority is facing the problem of starvation.

5.3.2.2 CHOPPER

An intelligent QoS-aware autonomic resource management approach for cloud computing (CHOPPER) framework ([Gill et al., 2017](#)) is used for resource provisioning in which: (i) clustering of workloads is done through workload priority, (ii) k-means based clustering algorithm is used for clustering of workloads, and (iii) QoS requirements of clustered workloads are identified and resources are provisioned by resource provisioner based on their QoS requirements.

5.3.3 Performance Analysis

5.3.3.1 Resource Utilization by Workload

Resource utilization by each workload is obtained through following equation used in CHOPPER (Gill et al., 2017).

$$RU = \sum_{i=1}^m \frac{RC_{(i)} - ET_{(i)}}{RC_{(i)}} \quad (5.2)$$

Where RU is resource utilization by each workloads in ms, m is number of workloads, $RC_{(i)}$ is resource completion request, and ET is execution time.

5.3.3.2 Execution Time

Execution time analysis obtain through Equ. 4.1, which is used in CHOPPER (Gill et al., 2017).

5.3.3.3 SLA Violation Rate

The SLA_cost is obtained through SLA violation rate, SLA violation rate obtained through failure rate. First of all it obtain failure rate.

To find the failure rate sstem has to set threshold value which uses all the weight values of the workload. The mean of the weight of workloads is considered as threshold. The following conditions which is used in CHOPPER (Gill et al., 2017) are used to calculate failure rate.

if weight value < threshold

failure(R)=2;

else if weight value > threshold

failure(R)=1;

$$failure_rate = \frac{failure(R_i)}{n} \quad (5.3)$$

Where, $failure(R_i)$ is the failure rate of every workload and n is the number of workload.

SLA Violation Rate

SLA violation rate is obtain through following equation

$$SVR = failure_rate * \sum_{i=1}^n W_i \quad (5.4)$$

The Equ. 5.3, and Equ. 5.4 is used in CHOPPER (Gill et al., 2017) where, W_i is the weight of each workload.

5.3.3.4 Energy Consumption Rate

The energy consumption rate is obtain through Equ. 5.5 used in SOCCER (Singh et al., 2016).

$$EnC = (EnC_{max} - EnC_{min}) * U_{VMi} + EnC_{min} \quad (5.5)$$

Where EnC is energy consumption rate, EnC_{max} is the energy consumption at the peak load (or 100% utilization), and EnC_{min} is the minimum energy consumption in the active/idle mode (or as low as 1% utilization). In this research, EnC_{max} is 1 kWh and EnC_{min} is $0 < EnC_{min} < 0.05$.

5.3.4 Execution Cost

The Execution cost is obtain through Equ. 5.6 used in CHOPPER (Gill et al., 2017).

$$EC = c * ET \quad (5.6)$$

Where c is initial cost that is \$30, as mentioned in TABLE 3.3, EC is Execution Cost and ET is Execution Time.

5.3.5 Resource Cost

The Resource cost is obtain through Equ. 8.6 used in SOCCER (Singh et al., 2016).

$$RC = c * EnC \quad (5.7)$$

Where c is initial cost that is \$30, as mentioned in TABLE 3.3, RC is Resource Cost and EnC is Energy Consumption Rate.

5.3.6 SLA Cost

The SLA cost is obtained through Equ. 5.8 used in CHOPPER (Gill et al., 2017).

$$SLAC = SVR * ET \quad (5.8)$$

Where SVR is SLA violation rate obtained through Equ. 5.4, $SLAC$ is SLA Cost and ET is Execution Time.

5.3.7 Resource Utilization

The objective of this section is to *Maximize the resource utilization through self-optimization.*

Through self-optimization, it obtained optimal VM's, and analyse resource utilization based on Equ. 5.1 and compared SMART results with two existing approaches SOCCER(Singh et al., 2016) and CHOPPER(Gill et al., 2017). The resource utilization is obtained for different workload dataset and presented in FIGURE 5.1 (a).

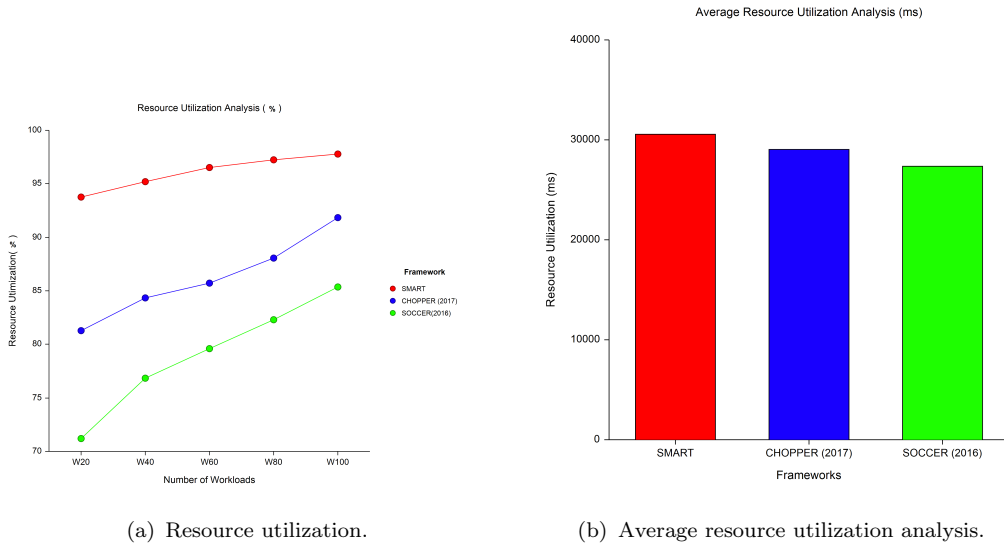


FIGURE 5.1: Resource utilization based on workloads

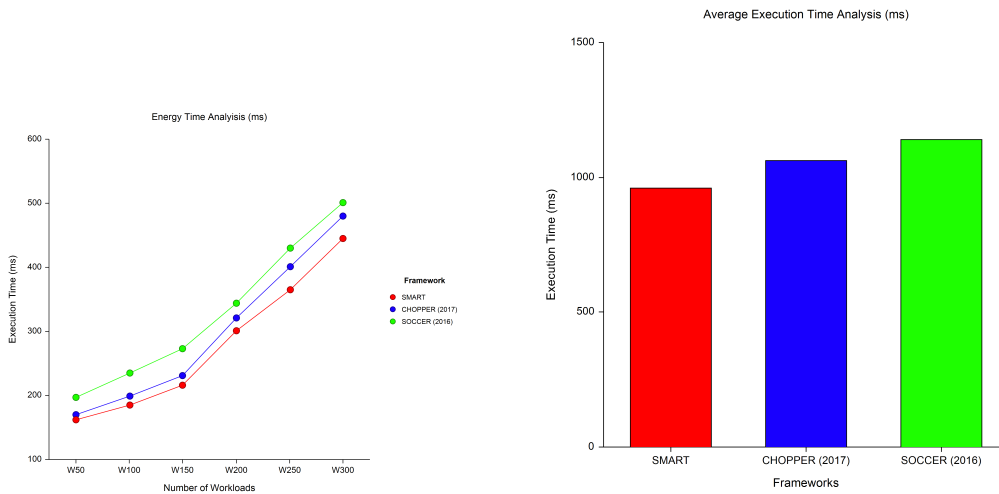
FIGURE 5.1 (a) & (b) shows the experimental results of SMART, CHOPPER, and SOCCER. In this total 30 test has been performed by increasing of 20 workloads in each iteration. As per the experimental results and observation, the minimum resource utilization for SMART, CHOPPER and SOCCER is as recorded 1857 ms, 1662 ms and 1424 ms respectively. The maximum resource utilization for SMART,

CHOPPER, and SOCCER is recorded as 59195 ms, 56103 ms, and 53368 ms respectively. Where as the average utilization of resources for SMART, CHOPPER, and SOCCER is recorded as 30523 ms, 29060 ms, and 27350 ms. The experimental observation for 30 test is the evidense that resource utilization rate of SMART is higher then CHOPPER and SOCCER. In FIGURE 5.1, it is clearly visible that the resource utilization of SMART is better than other two existing framework SOCCER and CHOPPER. SMART maximizes resource utilization by **5.03%**.

Hence the first sub-objective- *”Maximize the resource utilization through self-optimization”* is achieved successfully.

5.3.8 Execution Time

The execution time for SMART is recorded on the basis of succefully scheduling of VM to workloads submitted by cloud user. In this, prposed system is taken eight sample data for testing. It generate synthesis test data for average execution time, which is set of 50 to 1500 set of 30 workloads.The analysis of execution time and average execution time by SMART and other existing frameworks.



(a) Execution Time (ms) based on 50 to 300 Work- (b) Average Execution Time (ms) for 50 to 1500 loads.

FIGURE 5.2: Execution Time analysis based on number of workloads.

FIGURE 5.2 (a) & (b) shows the experimental results of SMART, CHOPPER, and SOCCER. In this total 30 test has been performed by increasing of 50 workloads in each iteration. As per the expemental results and observation, the minimum execution time for SMART, CHOPPER and SOCCER is recorded as 162 ms, 170 ms

and 197 ms respectively. The maximum execution time for SMART, CHOPPER, and SOCCER is recorded as 1824 ms, 2001 ms, and 2098 ms respectively. Where as the average execution time for SMART, CHOPPER, and SOCCER is recorded as 978 ms, 1070 ms, and 1135 ms. The experimental observation for 30 test is the evidense that execution time of SMART is lower then CHOPPER and SOCCER.

SMART reduces execution time by **8.6%**.

5.3.9 Energy Consumption Rate

The energy consumption is calculated in kilo watt hour (kWh) for SMART, CHOPPER and SOCCER for cloud resources (VM's). With the increasing number of workloads from 100 to 400, tests has been conducted and it is recorded that the value of energy consumption also increases while number of workloads are increases. The minimum value of energy consumption is 2.14 kWh at 100 workloads.

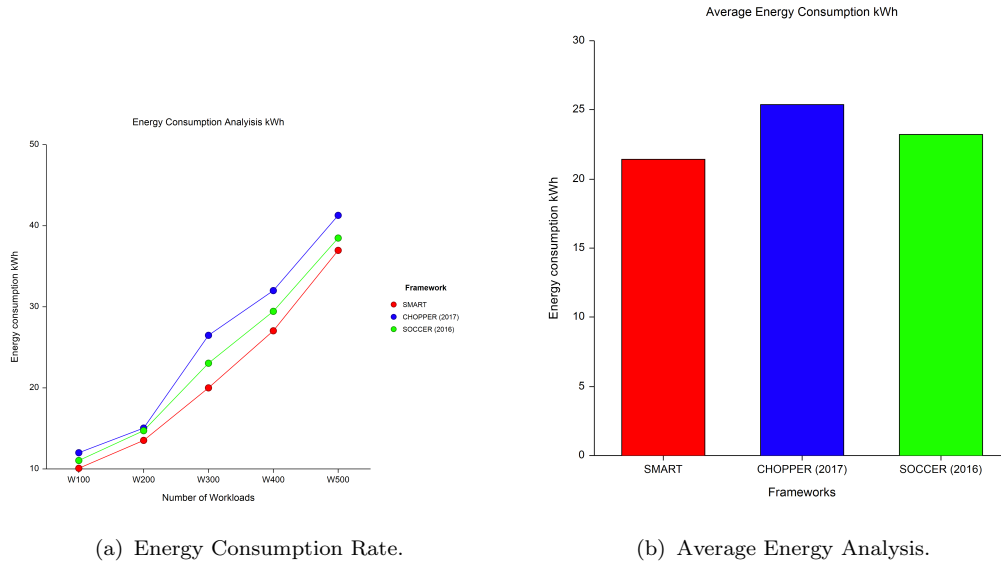
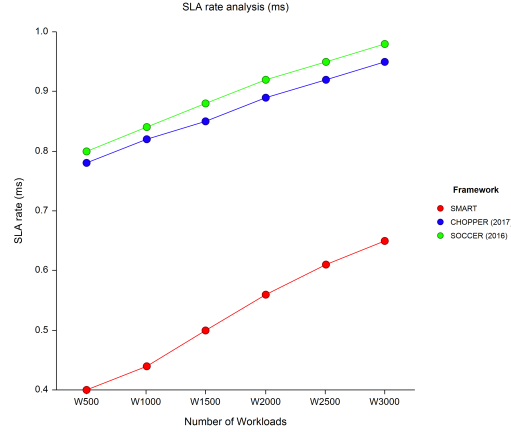


FIGURE 5.3: Energy Consumption Rate & Analysis based on Resources (VM).

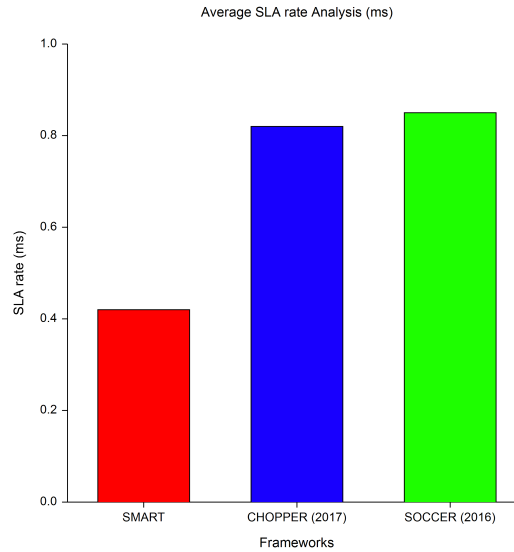
SMART performs better than CHOPPER and SOCCER in terms of energy consumption at different number of cloud resources (VM) as shown in FIGURE 5.3 (a). In FIGURE 5.3 (b) it is clearly shown that the average energy consumption in SMART is lesser than average energy consumption by CHOPPER and SOCCER.

5.3.10 SLA Violation Rate Analysis

The SLA violation rate analysis is obtained based on Number of Workloads. SMART process large dataset for computing SLA violation rate based on number of workloads, and results calculated through Equ. 5.3 and shown in FIGURE 5.4 for analysing its performance.



(a) SLA Violation Rate.



(b) Average SLA Rate Analysis.

FIGURE 5.4: SLA Violation Rate & Analysis based on workloads.

As per analysis shown in FIGURE 5.3 (a) & (b). The SLA violation rate is increasing while increasing number of workloads. The experimental results are recored

and analyse that SMART has less SLA violation rate as compared to SOCCER, and CHOPPER. SMART reduces SLA rate by **48.78%**.

5.3.11 Cost

The cost is obtained based on Energy consumption rate by Resources (VM), SLA Violation Rate with respect to Execution Time and Execution Cost based on Execution Time. The average cost is obtain for different workload dataset and number of Resources (VM). The following cost is evaluated for SMART:

1. Execution Cost,
2. Resource Cost, and
3. SLA Cost.

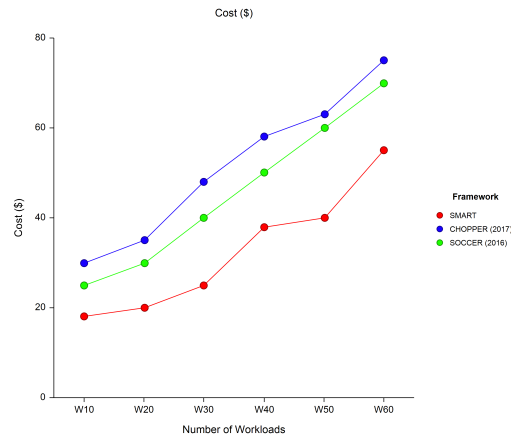
5.3.11.1 Execution Cost

It is defined as the product of total execution time to schedule of workload to optimal resources (VM) and price as per mentioned in TABLE 3.3. The execution cost is obtained through Equ. 5.5 and measured in dollars (\$) and results are shown FIGURE 5.4. As the number of workload increases, SMART performs better than SOCCER and CHOPPER.

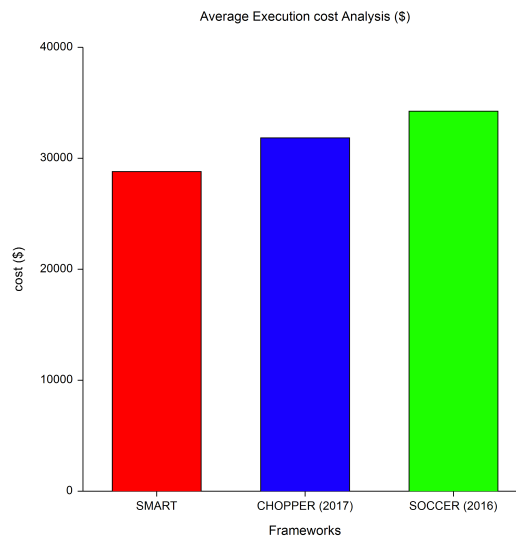
The cause is that SMART adjusts the resources at runtime according to the QoS requirements of workload. The execution cost is 4% lesser in SMART than CHOPPER and 6% lesser in SMART than SOCCER.

FIGURE 5.5 (a) & (b) shows the experimental results of SMART, CHOPPER, and SOCCER. In this total 30 test has been performed by increasing of 50 workloads in each iteration. As per the expemental results and observation, the minimum execution cost for SMART, CHOPPER and SOCCER is recorded as \$ 4860, \$ 5100 and \$ 5910 respectively. The maximum execution cost for SMART, CHOPPER, and SOCCER is recorded as \$ 54707, \$ 60027 and \$ 62326 respectively. Where as the average execution cost for SMART, CHOPPER, and SOCCER is recorded as \$ 29353, \$ 32112 and \$ 33761 respectively. The experimental observation for 30 test is the evidense that execution cost of SMART is lower then CHOPPER and SOCCER.

SMART reduces execution cost by **8.59%**.



(a) Execution Cost.



(b) Average Execution Cost Analysis.

FIGURE 5.5: Execution Cost & Analysis based on workloads.

5.3.11.2 Resource Cost

It is defined as the product of total energy consumption by resource (VM) and price as per mentioned in TABLE 3.3. The resource cost is obtained through Equ. 5.7 and measured in dollars (\$) and analysis is shown in FIGURE 5.6. Resource cost rises as shown in FIGURE 5.5.

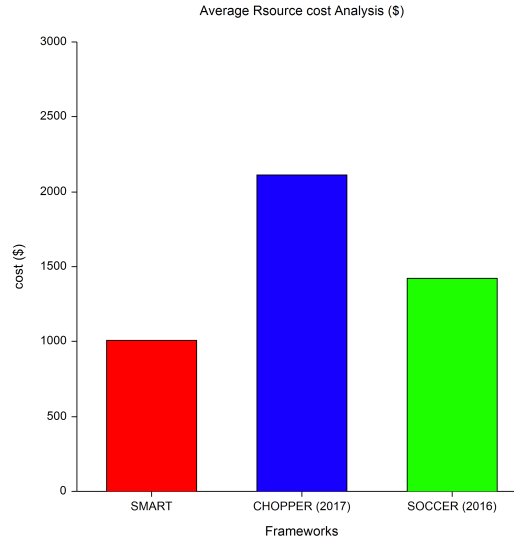


FIGURE 5.6: Resource Cost & Analysis based on workloads.

FIGURE 5.6 shows the experimental results of SMART, CHOPPER, and SOCCER. In this total 30 test has been performed by increasing of 4 VM's in each iteration. As per the expemental results and observation, the minimum resource cost for SMART, CHOPPER and SOCCER is recorded as \$ 64, \$ 201 and \$ 145 respectively. The maximum resource cost for SMART, CHOPPER, and SOCCER is recorded as \$ 1946, \$ 3950 and \$ 2921 respectively. Where as the average resource cost for SMART, CHOPPER, and SOCCER is recorded as \$ 1005, \$ 2075 and \$ 1533 respectively. The experimental observation for 30 test is the evidense that resource cost of SMART is lower then CHOPPER and SOCCER.

5.3.11.3 SLA Cost

It is defined as the product of total SLA violation rate by workloads and total execution time. The SLA cost is obtained through Equ. 5.8 and measured in dollars (\$). As the number of workloads increases, SMART performs better than SOCCER and CHOPPER. The cause is that SMART adjusts the resources at runtime according to the QoS requirements of workload. SLA cost rises as shown in FIGURE 5.7

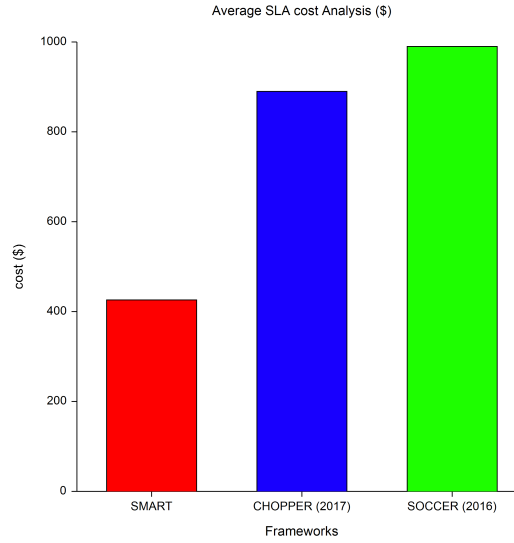


FIGURE 5.7: SLA Cost & Analysis based on workloads.

5.3.12 Analysis

The analyse of Resource Utilization, Execution Time, Average Cost, Energy Efficiency, and SLA Violation Rate for SMART, SOCCER, and CHOPPER. In all respect it is recorded that SMART perform better than the existing framework in the mentioned simulation environment. The average of all parameters evaluvated and presented in TABLE 5.1.

TABLE 5.1: Analysis

Parameters	SMART	CHOPPER	SOCCER	Analysis
VM Utilization	30523 ms	29060 ms	27350 ms	Maximize 5%
Execution Time	978 ms	1070 ms	1135 ms	Minimize 8%
SLA Violation	0.420 ms	0.850 ms	0.820ms	Minimize 48%
Execution Cost	\$29353	\$32112	\$33776	Minimize 8.60%

The above analysis has been carried out based on the results, which is simulated in CloudSim 3.0, for SMART, CHOPPER and SOCCER.

Hense the third sub-objective -

”Attain the better system by incorporating an efficient and intelligent resource scheduling algorithm in cloud computing” is achieved succesfully.

5.3.13 Implementation in AWS

The proposed research work SMART is implemented in Amazon Web Services AWS as per comments received from SRC panel members. In AWS, 6 Instances. Main algorithm is written in ALOscheduler (name of server), workloads are stored in second one and client application is stored in third instance as presented in FIGURE 5.8.

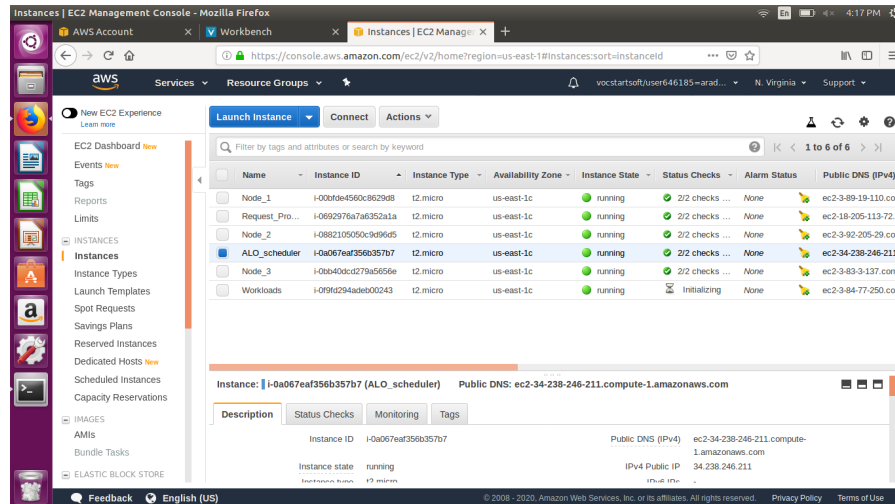


FIGURE 5.8: EC2 Dashboard

FIGURE 5.9 and 5.10 presented ALOscheduler and client application terminal screenshots. In this, the initial status of ALOscheduler EC2 instance is nil. Now this instance is ready to perform scheduling algorithm.

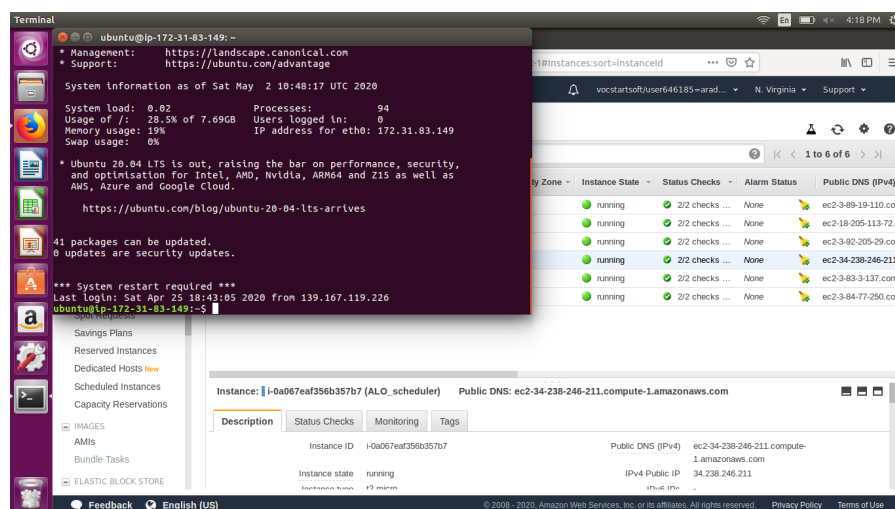


FIGURE 5.9: ALOscheduler EC2 Instance

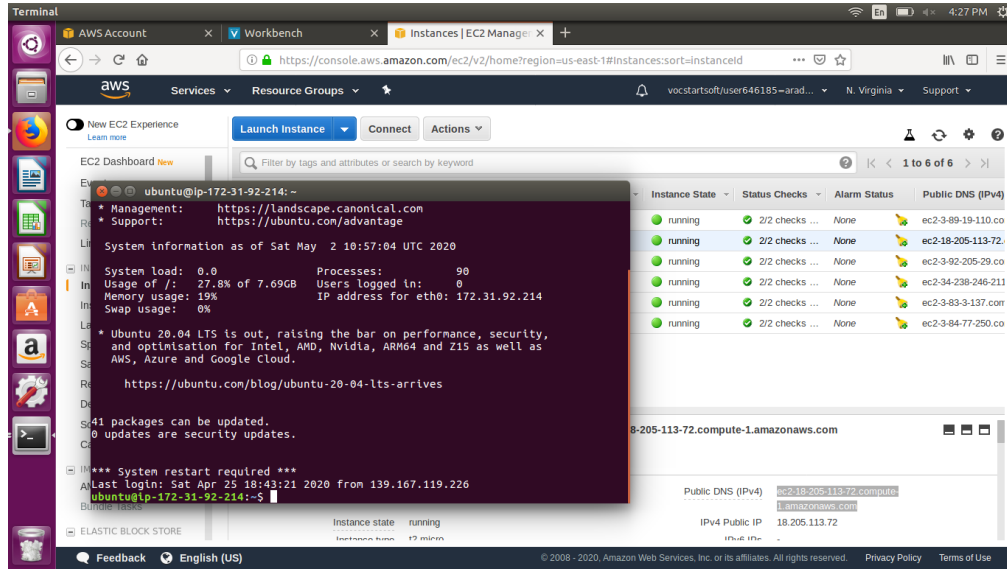


FIGURE 5.10: Client application EC2 Instance

FIGURE 5.11 presented all three terminals where ALOScheduler, workloads and client application are installed.

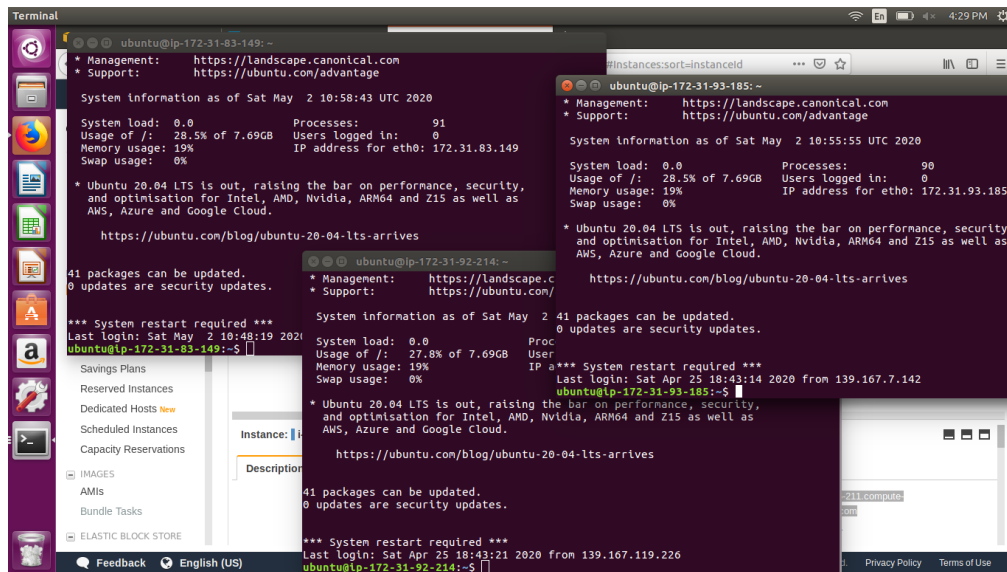
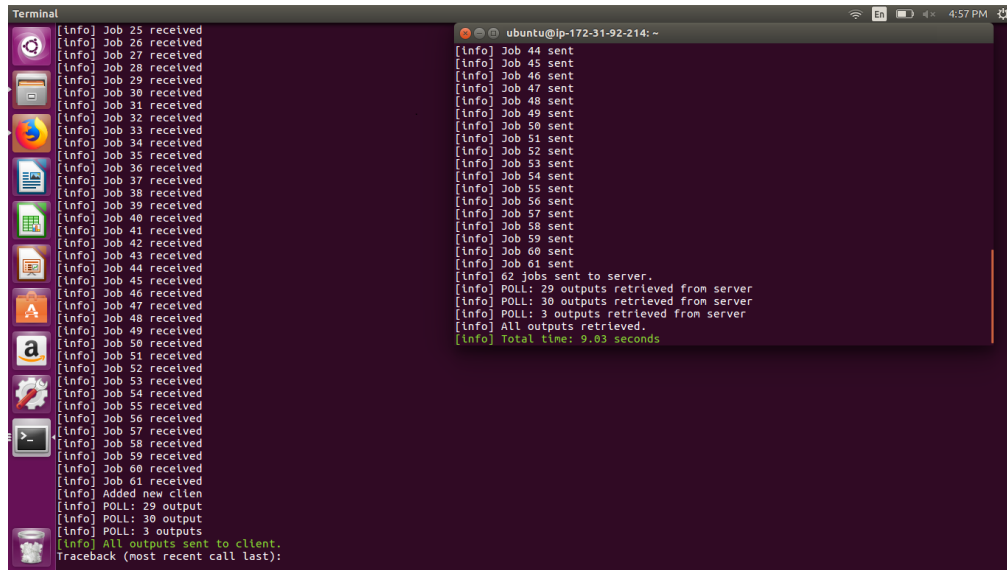


FIGURE 5.11: ALOScheduler, Client, and Worker EC2 Instance

FIGURE 5.12 presented the execution of 62 workloads in AWS environment, and it takes 9.03 seconds to compute the workloads in 5 instances.

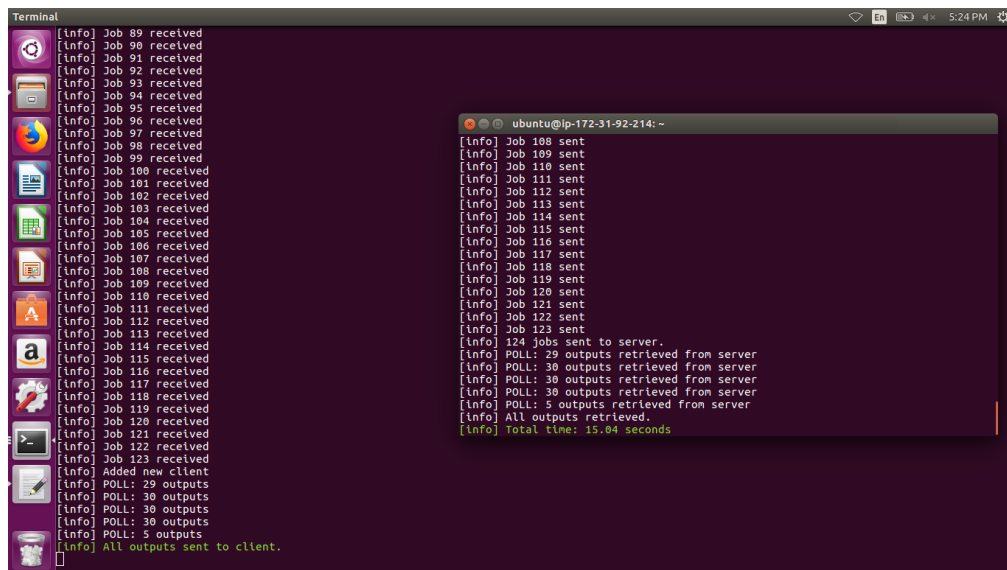


```
[Info] Job 25 received
[Info] Job 26 received
[Info] Job 27 received
[Info] Job 28 received
[Info] Job 29 received
[Info] Job 30 received
[Info] Job 31 received
[Info] Job 32 received
[Info] Job 33 received
[Info] Job 34 received
[Info] Job 35 received
[Info] Job 36 received
[Info] Job 37 received
[Info] Job 38 received
[Info] Job 39 received
[Info] Job 40 received
[Info] Job 41 received
[Info] Job 42 received
[Info] Job 43 received
[Info] Job 44 received
[Info] Job 45 received
[Info] Job 46 received
[Info] Job 47 received
[Info] Job 48 received
[Info] Job 49 received
[Info] Job 50 received
[Info] Job 51 received
[Info] Job 52 received
[Info] Job 53 received
[Info] Job 54 received
[Info] Job 55 received
[Info] Job 56 received
[Info] Job 57 received
[Info] Job 58 received
[Info] Job 59 received
[Info] Job 60 received
[Info] Job 61 received
[Info] Added new client
[Info] POLL: 29 output
[Info] POLL: 30 output
[Info] POLL: 3 outputs
[Info] All outputs sent to client.
Traceback (most recent call last):

ubuntu@ip-172-31-92-214: ~
[Info] Job 44 sent
[Info] Job 45 sent
[Info] Job 46 sent
[Info] Job 47 sent
[Info] Job 48 sent
[Info] Job 49 sent
[Info] Job 50 sent
[Info] Job 51 sent
[Info] Job 52 sent
[Info] Job 53 sent
[Info] Job 54 sent
[Info] Job 55 sent
[Info] Job 56 sent
[Info] Job 57 sent
[Info] Job 58 sent
[Info] Job 59 sent
[Info] Job 60 sent
[Info] Job 61 sent
[Info] 62 jobs sent to server.
[Info] POLL: 29 outputs retrieved from server
[Info] POLL: 30 outputs retrieved from server
[Info] POLL: 3 outputs retrieved from server
[Info] All outputs retrieved.
[Info] Total time: 9.03 seconds
```

FIGURE 5.12: Execution of workloads

FIGURE 5.13 presents the execution of 125 workloads in AWS environments, and it takes 15.04 seconds to execute all the workloads.



```
[Info] Job 89 received
[Info] Job 90 received
[Info] Job 91 received
[Info] Job 92 received
[Info] Job 93 received
[Info] Job 94 received
[Info] Job 95 received
[Info] Job 96 received
[Info] Job 97 received
[Info] Job 98 received
[Info] Job 99 received
[Info] Job 100 received
[Info] Job 101 received
[Info] Job 102 received
[Info] Job 103 received
[Info] Job 104 received
[Info] Job 105 received
[Info] Job 106 received
[Info] Job 107 received
[Info] Job 108 received
[Info] Job 109 received
[Info] Job 110 received
[Info] Job 111 received
[Info] Job 112 received
[Info] Job 113 received
[Info] Job 114 received
[Info] Job 115 received
[Info] Job 116 received
[Info] Job 117 received
[Info] Job 118 received
[Info] Job 119 received
[Info] Job 120 received
[Info] Job 121 received
[Info] Job 122 received
[Info] Job 123 received
[Info] Added new client
[Info] POLL: 29 outputs
[Info] POLL: 30 outputs
[Info] POLL: 30 outputs
[Info] POLL: 30 outputs
[Info] POLL: 5 outputs
[Info] All outputs sent to client.

ubuntu@ip-172-31-92-214: ~
[Info] Job 108 sent
[Info] Job 109 sent
[Info] Job 110 sent
[Info] Job 111 sent
[Info] Job 112 sent
[Info] Job 113 sent
[Info] Job 114 sent
[Info] Job 115 sent
[Info] Job 116 sent
[Info] Job 117 sent
[Info] Job 118 sent
[Info] Job 119 sent
[Info] Job 120 sent
[Info] Job 121 sent
[Info] Job 122 sent
[Info] Job 123 sent
[Info] 124 jobs sent to server.
[Info] POLL: 29 outputs retrieved from server
[Info] POLL: 30 outputs retrieved from server
[Info] POLL: 30 outputs retrieved from server
[Info] POLL: 30 outputs retrieved from server
[Info] POLL: 5 outputs retrieved from server
[Info] All outputs retrieved.
[Info] Total time: 15.04 seconds
```

FIGURE 5.13: Execution of workloads

The total 4000 task has been submitted and results are recorded. The following FIGURE 5.14 presented the analysis of execution time. While increasing of number of tasks, the execution time is also increases.

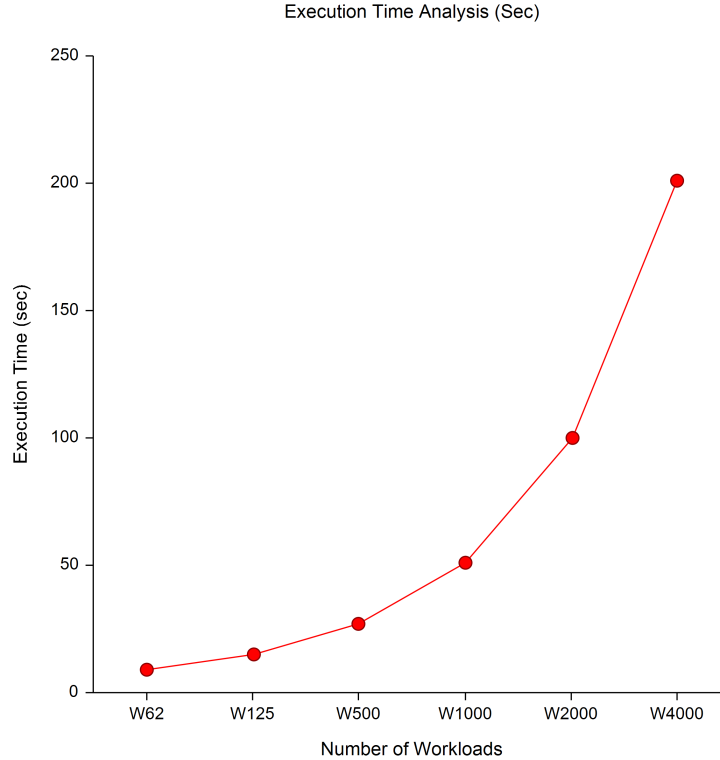


FIGURE 5.14: Execution Time Analysis in AWS environment

5.4 Summary

SMART optimized resources and find optimal resources and separate fault VM's in this work, so best resources which has high fitness value for better performance is produce for scheduling. On other hand, it is also computed and identified duplicate workloads to avoid accidentally repeatable workloads and seperated it from submission list. Both resources and workloads are optimal absed on its fitness value and user priority for scheduling. SMART approach round-robbin scheduling algorithm to schedule available resources to workloads. The proposed system is implemented and simulated in the Amazon Web Service AWS and cloudsim 3.0 environment. The proposed optimization method is to be finds the optimal VM's as discussed. In continuous, threshold for each VM parameters obtain and identified the VM's which have low utilization capacity in terms of CPU, RAM, and Bandwidth and separate from resource pool. These VM's are observed as fault VM's which is to be calucted through faut-tolerant procedure as discussed. The objective of proposed system is to produce an efficient autonomic resource management technique in cloud, which named as SMART to improve the efficiency of the existing resource management technique. The SMART algorithm is a hybrid approach to manage

resource with its best utilization value. SMART successfully identified workloads as per user priority and optimal resource for best utilization and schedule VM's as per QoS.

SMART compute different performance metrics based on equations and it is required that SMART perform better and satisfy all the sub-objective of this work, which results that the objective of this work-

”To design a resource management system to improve the fault tolerance mechanism, self-adaptability, and maximize resource utilization in cloud computing”

is achieved.

Chapter 6

Conclusions and Future Work

Resource management in cloud computing is the ability to manage resources according to the user's need in the cloud. The operating cost of the cloud services is mostly dependent upon resource utilization. Also, customer satisfaction is one of the major highlighting metrics to trust of cloud services subjected to SLA violation rate. Due to the increasing demand for services, a large number of servers need to be installed to fulfill the customer requirements, which increases the energy consumption, which affects the operating cost to the service provider and customer as well. To overcome this issue, an efficient autonomic resource management technique needs to be applied, that is self-managing the resources without human interventions. We propose an efficient self-management aware autonomic resource management technique in cloud SMART to reduce the average cost with customer satisfaction. In this, to identify the faulty VM's, we implemented an automated self-healing cloud computing framework for resource scheduling, which finds the fault VM's from the resource pool to reject/avoid to allocate the workload submitted by cloud user, which improves the performance in terms of execution time, and average cost. To continue this, we optimized resources based on modified ALO, which minimizes the energy consumption and SLA violation rate so that resources can be efficiently utilized, and resource utilization can be maximized.

The experimental observations for different performance parameters are recorded for proposed research work SMART and existing RMS, where it has been recorded that the average resource utilization by SMART is increased by 5%, the average execution time is minimized by SMART is 8%, the average SLA violation rate is minimized by SMART is 48%, and the average execution cost is minimized by SMART is 8.60%.

The experimental results are compared with Soccer and Chopper and it is recorded that SMART is more efficient. To manage the workloads efficiently, we applied self-protection in SMART, which helps to identify the malicious workloads submitted

by the cloud user. The objective of SMART is to attain a better resource management system by incorporating an efficient and intelligent resource management algorithm in cloud computing. The efficiency metric for SMART computing SLA violation rate is based on workloads, execution cost, energy consumption rate, and energy efficiency analysis, execution time, and resource utilization. SMART is simulated in Cloudsim toolkit and compares with existing systems. SMART reduces SLA violation rate based on workloads, reduces execution cost, energy consumption rate, and increases energy efficiency as compared with existing systems. Hence, SMART is an efficient autonomic resource management technique in cloud computing.

Future Work

There is always some scope of improvements present in each research. Similarly, in SMART we have identified some scope which can be extended in the future to improve the quality of this research:

1. **Scaling and migration of VM's:** The SMART work on a fixed number of VM's, further it can be extended to dynamic increments of VM or it can use of other service providers VM.
2. Machine learning concept can be utilized for the real-time environment to get better efficiency.
3. Energy efficiency by calculation of optimal threshold value for energy consumption rate by applying machine learning and linear regression method.
4. SMART does not heal the resources, further, it can be improved by adopting a self-healing mechanism.
5. Self-configuration is one more characteristic of autonomic computing, which is responsible for re-installation of missing components, which can be included in SMART for further enhancements.

Appendix A

List of Publications

Paper Published in SCI/ SCOPUS/ ESCI indexed Journals:

1. Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha. 2020. Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. *Software: Practice and Experience*.4(1). PP: 1-20. **SCI**
<https://doi.org/10.1002/spe.2810>
2. Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha. 2020. Cloud Resource Optimization System based on Time and Cost. *International Journal of Mathematical, Engineering and Management Sciences*.5(4). PP: 758-768. **SCOPUS Q2**
<http://doi.org/10.33889/IJMEMS.2020.5.4.060>
3. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Self-characteristics based Energy-Efficient Resource Scheduling for Cloud, *Procedia Computer Science*, Vol 152. PP: 204-211. **SCOPUS**
DOI: 10.1016/j.procs.2019.05.044
4. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Energy-Aware Autonomic Resource Scheduling Framework for Cloud. *International Journal of Mathematical, Engineering and Management Sciences*.4(1). PP: 41-55. **SCOPUS Q2**
<http://www.ijmems.in/assets/4-ijmems-18-234-vol.1-no.1-41-55-2019.pdf>
DOI: 10.33889/IJMEMS.2019.4.1-004
5. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Design of Self-Management Aware Autonomic Resource Scheduling Scheme in Cloud. *International Journal of Computer Information Systems and Industrial Management Applications*. Vol (11), PP: 170-177.**SCOPUS**

Q3, IET Inspec. MirLabs USA.

http://www.mirlabs.net/ijcisim/regular_papers_2019/IJCISIM_17.pdf

6. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. An Automated Self-Healing Cloud Computing Framework for Resource Scheduling. *International Journal of Grid and High Performance Computing*. **Accepted for Vol (13), Issue (2),pp: 47-64. SCOPUS Q2, ESCI, IET Inspec. IGI Global. USA**
7. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Sla-Based Autonomic Cloud Resource Managementframework By Antlion Optimization Algorithm. *International Journal of Innovative Technology and Exploring Engineering*. 8(4).119-123. **SCOPUS**.
<https://www.ijitee.org/wp-content/uploads/papers/v8i4/D2659028419.pdf>
8. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. A Self-Optimization Based Virtual Machine Scheduling to Workloads in Cloud Computing Environment. *International Journal of Engineering and Advanced Technology*. 8(4). 91-96. **SCOPUS**.
<https://www.ijeat.org/wp-content/uploads/papers/v8i4c/D24270484C19.pdf>

Paper presented in SCOPUS indexed International Conferences:

1. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2018. Credential and Security Issues of Cloud Service Models. in *proceesing of 2016 IEEE conference on Next Generation Computing Technologies* Publisher **IEEE SCOPUS**.
<https://ieeexplore.ieee.org/document/7877536>
DOI: 10.1109/NGCT.2016.7877536
2. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2018. Autonomic Cloud Resource Management. *Fifth IEEE international Conference on Parallel, Distributed and Grid Computing(PDGC)*. PP: 138-143. **IEEE SCOPUS**.
DOI: <https://ieeexplore.ieee.org/document/8745977>
DOI: 10.1109/PDGC.2018.8745977
3. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2018. SLA-aware resource optimization based on time and cost in cloud. *4th International Conference on Next Generation Computing Technologies NGCT 2018. UPES, Dehradun. India*. Presented 20-21 Nov. 2018. **Springer SCOPUS**.

Patent:

1. Dewangan, B.K, Choudhury, T., Agarwal, A. Et al. (2020). An efficient and cost effective system design for low cost cloud resource management [Govt. of India 202011023216 A]. Indian Patents.

References

- Abdullahi, M., Ngadi, M. A., et al. (2016). Symbiotic organism search optimization based task scheduling in cloud computing environment. *Future Generation Computer Systems*, 56, 640–650.
- Addis, B., Ardagna, D., Panicucci, B., & Zhang, L. (2010). Autonomic management of cloud service centers with availability guarantees. In *Cloud computing (cloud), 2010 IEEE 3rd international conference on* (pp. 220–227).
- Alex, Yamini, R., & Germanus, M. (2017). Comparison of resource optimization algorithms in cloud computing. In (pp. 847–855).
- AlJahdali, H., Albatli, A., Garraghan, P., Townend, P., Lau, L., & Xu, J. (2014). Multi-tenancy in cloud computing. In *Service oriented system engineering (sose), 2014 IEEE 8th international symposium on* (pp. 344–351).
- Alkhanak, E. N., & Lee, S. P. (2018). A hyper-heuristic cost optimisation approach for scientific workflow scheduling in cloud computing. *Future Generation Computer Systems*.
- Alsadie, D., Tari, Z., Alzahrani, E. J., & Zomaya, A. Y. (2018). Dynamic resource allocation for an energy efficient vm architecture for cloud computing. In *Proceedings of the Australasian computer science week multiconference* (p. 16).
- Al Salami, N. M. (2009). Ant colony optimization algorithm. *UbiCC Journal*, 4(3), 823–826.
- Altmann, J., & Kashef, M. M. (2014). Cost model based service placement in federated hybrid clouds. *Future Generation Computer Systems*, 41, 79–90.
- An, B., Lesser, V., Irwin, D., & Zink, M. (2010). Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems: volume 1-volume 1* (pp. 981–988).
- Antonescu, A. F., & Braun, T. (2016). Simulation of sla-based vm-scaling algorithms for cloud-distributed applications. *Future Generation Computer Systems*, 54, 260–273.

- Anuj Kumar Yadav, M., & Ritika. (2017). The issues of energy efficiency in cloud computing based data centers. *Biosc.Biotech.Res.Comm.*, 12(2), 1–10.
- Arianyan, E., Taheri, H., & Sharifian, S. (2015). Novel energy and sla efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Computers & Electrical Engineering*, 47, 222–240.
- Bansal, N., Maurya, A., Kumar, T., Singh, M., & Bansal, S. (2015). Cost performance of qos driven task scheduling in cloud computing. *Procedia Computer Science*, 57, 126–130.
- Banu, M. U., & Saravanan, K. (2014). Optimizing the cost for resource subscription policy in iaas cloud. *arXiv preprint arXiv:1402.2491*.
- Beloglazov, A., & Buyya, R. (2016, June 7). *System, method and computer program product for energy-efficient and service level agreement (sla)-based management of data centers for cloud computing*. Google Patents. (US Patent 9,363,190)
- Bing, L., Song, A. M., & Song, J. (2012). A distributed qos-constraint task scheduling scheme in cloud computing environment: model and algorithm. *Advances in information sciences and service sciences*, 4(5), 283–291.
- Bittencourt, L. F., & Madeira, E. R. M. (2011). Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2(3), 207–227.
- Bruneo, D., Longo, F., Ghosh, R., Scarpa, M., Puliafito, A., & Trivedi, K. S. (2015). Analytical modeling of reactive autonomic management techniques in iaas clouds. In *Cloud computing (cloud), 2015 ieee 8th international conference on* (pp. 797–804).
- Buyya, R., Pandey, S., & Vecchiola, C. (2009). Cloudbus toolkit for market-oriented cloud computing. In *Ieee international conference on cloud computing* (pp. 24–44).
- Cai, X., Feng, L., Ping, L., Lei, J., & Zhiping, J. (2017). Sla-aware energy-efficient scheduling scheme for hadoop yarn. *The Journal of Supercomputing*, 73(8), 3526–3546.
- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1), 23–50.
- Casalicchio, E., Menascé, D. A., & Aldhalaan, A. (2013). Autonomic resource provisioning in cloud systems with availability goals. In *Proceedings of the 2013 acm cloud and autonomic computing conference* (p. 1).

- Chaisiri, S., Lee, B. S., & Niyato, D. (2012). Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5(2), 164–177.
- Changtian, Y., & Jiong, Y. (2012). Energy-aware genetic algorithms for task scheduling in cloud computing. In *Chinagrid annual conference (chinagrid), 2012 seventh* (pp. 43–48).
- Choi, Y., & Lim, Y. (2016). Optimization approach for resource allocation on cloud computing for iot. *International Journal of Distributed Sensor Networks*, 2016, 23.
- Convolbo, M. W., & Chou, J. (2016). Cost-aware dag scheduling algorithms for minimizing execution cost on cloud resources. *The Journal of Supercomputing*, 72(3), 985–1012.
- Coutinho, R., Drummond, & Frota, Y. (2013). Optimization of a cloud resource management problem from a consumer perspective. In *European conference on parallel processing* (pp. 218–227).
- Dai, Y., Lou, Y., & Lu, X. (2015). A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing. In *Intelligent human-machine systems and cybernetics (ihmsc), 2015 7th international conference on* (Vol. 2, pp. 428–431).
- Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. In *Foundations of computational intelligence volume 3* (pp. 23–55). Springer.
- Delimitrou, C., & Kozyrakis, C. (2014). Quasar: resource-efficient and qos-aware cluster management. *ACM SIGPLAN Notices*, 49(4), 127–144.
- Deng, J., Huang, S. C., Han, Y. S., & Deng, J. H. (2010). Fault-tolerant and reliable computation in cloud computing. In *Globecom workshops (gc wkshps), 2010 ieee* (pp. 1601–1605).
- Dewangan, B. K., Agarwal, A., Marryboyina, V., & Pasricha, A. (2018). Resource scheduling in cloud: A comparative study. *International Journal of Computer Sciences and Engineering*, 6(8), 168–173.
- Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: issues and challenges. In *Advanced information networking and applications (aina), 2010 24th ieee international conference on* (pp. 27–33).
- Ding, J., Zhang, Z., Ma, R. T., & Yang, Y. (2016). Auction-based cloud service differentiation with service level objectives. *Computer Networks*, 94, 231–249.

- Dou, W., Xu, X., Meng, S., Zhang, X., Hu, C., Yu, S., & Yang, J. (2017). An energy-aware virtual machine scheduling method for service qos enhancement in clouds over big data. *Concurrency and Computation: Practice and Experience*, 29(14), e3909.
- Egwutuoha, I. P., Chen, S., Levy, D., & Selic, B. (2012). A fault tolerance framework for high performance computing in cloud. In *Cluster, cloud and grid computing (ccgrid), 2012 12th ieee/acm international symposium on* (pp. 709–710).
- Elster, A. C., Tzovaras, D., Petcu, D., & Morrison, J. P. (2016). Cloudlightning: A framework for a self-organising and self-managing heterogeneous cloud.
- Fang, L. X., Zhan, Z. H., Jing, D., & Chen, W. N. (2014). Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (pp. 41–48).
- Fang, Y., Chen, Q., & Xiong, N. (2018). A multi-factor monitoring fault tolerance model based on a gpu cluster for big data processing. *Information Sciences*.
- Fargo, F., Tunc, C., Al-Nashif, Y., Akoglu, A., & Hariri, S. (2014). Autonomic workload and resources management of cloud computing services. In *Cloud and autonomic computing (iccac), 2014 international conference on* (pp. 101–110).
- Gai, K., Qiu, M., Zhao, H., Tao, L., & Zong, Z. (2016). Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications*, 59, 46–54.
- Gan, C. Z., Du, K.-J., Zhan, Z.-H., & Zhang, J. (2015). Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In *Evolutionary computation (cec), 2015 ieee congress on* (pp. 708–714).
- Gang, Z. (2014). Cost-aware scheduling algorithm based on pso in cloud computing environment. *International Journal of Grid and Distributed Computing*, 7(1), 33–42.
- Gao, G., Hu, H., Wen, Y., & Westphal, C. (2017). Resource provisioning and profit maximization for transcoding in clouds: A two-timescale approach. *IEEE Transactions on Multimedia*, 19(4), 836–848.
- Garg, S. K., Toosi, A. N., Gopalaiyengar, S. K., & Buyya, R. (2014). Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter. *Journal of Network and Computer Applications*, 45, 108–120.
- Ghasemi, S., Meybodi, M. R., Fooladi, M. D. T., & Rahmani, A. M. (2017). A cost-aware mechanism for optimized resource provisioning in cloud computing. *Cluster Computing*, 1–14.

- Ghobaei Arani, M., Jabbehdari, S., & Pourmina, M. A. (2016). An autonomic approach for resource provisioning of cloud services. *Cluster Computing*, 19(3), 1017–1036.
- Ghobaei Arani, M., Jabbehdari, S., & Pourmina, M. A. (2018). An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Generation Computer Systems*, 78, 191–210.
- Gill, S. S., Chana, I., Singh, M., & Buyya, R. (2017). Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing. *Cluster Computing*, 1–39.
- Guo, C., & Yu, J. (2005). Particle swarm optimization algorithm. *INFORMATION AND CONTROL-SHENYANG-*, 34(3), 318.
- Guo, S., Xiao, B., Yang, Y., & Yang, Y. (2016). Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In *Infocom 2016-the 35th annual ieee international conference on computer communications, ieee* (pp. 1–9).
- Hoenisch, P., Hochreiner, C., Schuller, D., Schulte, S., Mendling, J., & Dustdar, S. (2015). Cost-efficient scheduling of elastic processes in hybrid clouds. In *Cloud computing (cloud), 2015 ieee 8th international conference on* (pp. 17–24).
- Hong, Z., Jiang, H., Li, B., Liu, F., Vasilakos, A. V., & Liu, J. (2016). A framework for truthful online auctions in cloud computing with heterogeneous user demands. *IEEE Transactions on Computers*, 65(3), 805–818.
- Iyer, G. N., & Veeravalli, B. (2011). On the resource allocation and pricing strategies in compute clouds using bargaining approaches. In *Networks (icon), 2011 17th ieee international conference on* (pp. 147–152).
- Jala, J., & Ramchand, R. K. (2019). Qos-based technique for dynamic resource allocation in cloud services. In *International conference on computer networks and communication technologies* (pp. 65–73).
- Jamshidi, P., Ahmad, A., & Pahl, C. (2014). Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems* (pp. 95–104).
- Jamshidi, P., Pahl, C., & Mendonça, N. C. (2016). Managing uncertainty in autonomic cloud elasticity controllers. *IEEE Cloud Computing*(3), 50–60.
- Jansen, W. A. (2011). Cloud hooks: Security and privacy issues in cloud computing. In *2011 44th hawaii international conference on system sciences* (pp. 1–10).

- Jayadivya, S., Nirmala, J. S., & Bhanu, M. S. S. (2012). Fault tolerant workflow scheduling based on replication and resubmission of tasks in cloud computing. *International Journal on Computer Science and Engineering*, 4(6), 996.
- Jayaswal, K., & Shah, D. (2015). Cloud computing black book.
- Jhawar, R., Piuri, V., & Santambrogio, M. (2012). A comprehensive conceptual system-level approach to fault tolerance in cloud computing. In *Systems conference (syscon), 2012 ieee international* (pp. 1–5).
- Jin, L., Song, W., & Zhuang, W. (2018). Auction-based resource allocation for sharing cloudlets in mobile cloud computing. *IEEE Transactions on Emerging Topics in Computing*, 6(1), 45–57.
- Jing, L., Zhou, J., & Buyya, R. (2015). Software rejuvenation based fault tolerance scheme for cloud applications. In *Cloud computing (cloud), 2015 ieee 8th international conference on* (pp. 1115–1118).
- Jixian, Z., Xie, N., Zhang, X., & Li, W. (2018). An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost. *Future Generation Computer Systems*, 89, 286–299.
- Kamal, K., & Kemafor, A. (2010). Scheduling hadoop jobs to meet deadlines. In *Cloud computing technology and science (cloudcom), 2010 ieee second international conference on* (pp. 388–392).
- Kang, D. K., Kim, S. H., Youn, C. H., & Chen, M. (2014). Cost adaptive workflow scheduling in cloud computing. In *Proceedings of the 8th international conference on ubiquitous information management and communication* (p. 65).
- Kansal, N. J., & Chana, I. (2015). Artificial bee colony based energy-aware resource utilization technique for cloud computing. *Concurrency and Computation: Practice and Experience*, 27(5), 1207–1225.
- Kaur, P., & Mehta, S. (2017). Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm. *Journal of Parallel and Distributed Computing*, 101, 41–50.
- Koch, F., Assunção, M. D., Cardonha, C., & Netto, M. A. (2016). Optimising resource costs of cloud computing for education. *Future Generation Computer Systems*, 55, 473–479.
- Kohne, A., Pasternak, D., Nagel, L., & Spinczyk, O. (2016). Evaluation of slab-based decision strategies for vm scheduling in cloud data centers. In *Proceedings of the 3rd workshop on crosscloud infrastructures & platforms* (p. 6).
- Kong, W., Lei, Y., & Ma, J. (2016). Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism. *Optik-International Journal for Light and Electron Optics*, 127(12), 5099–5104.

- Latiff, M. S. A., et al. (2017). A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness. *Applied Soft Computing*, 61, 670–680.
- Lee, Y. C., Wang, C., Zomaya, A. Y., & Zhou, B. B. (2010). Profit-driven service request scheduling in clouds. In *Cluster, cloud and grid computing (ccgrid), 2010 10th ieee/acm international conference on* (pp. 15–24).
- Licklider, J. C., & Taylor, R. W. (1968). The computer as a communication device. *Science and technology*, 76(2), 1–3.
- Lin, W. Y., Lin, G.-Y., & Wei, H. Y. (2010). Dynamic auction mechanism for cloud resource allocation. In *Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing* (pp. 591–592).
- Linlin, W., Garg, S. K., & Buyya, R. (2011). Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Proceedings of the 2011 11th ieee/acm international symposium on cluster, cloud and grid computing* (pp. 195–204).
- Liu, Jin, H., Chen, J., Liu, X., Yuan, D., & Yang, Y. (2010). A compromised-time-cost scheduling algorithm in swindow-c for instance-intensive cost-constrained workflows on a cloud computing platform. *The International Journal of High Performance Computing Applications*, 24(4), 445–456.
- Malawski, M., Juve, G., Deelman, E., & Nabrzyski, J. (2015). Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. *Future Generation Computer Systems*, 48, 1–18.
- Malik, S., & Huet, F. (2011). Adaptive fault tolerance in real time cloud computing. In *Services (services), 2011 ieee world congress on* (pp. 280–287).
- Marzband, M., Azarnejadian, F., Savaghebi, M., & Guerrero, J. M. (2017). An optimal energy management system for islanded microgrids based on multiperiod artificial bee colony combined with markov chain. *IEEE Systems Journal*, 11(3), 1712–1722.
- Meena, J., Kumar, M., & Vardhan, M. (2016). Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint. *IEEE Access*, 4, 5065–5082.
- Mehta, S., & Kaur, P. (2019). Scheduling data intensive scientific workflows in cloud environment using nature inspired algorithms. In *Nature-inspired algorithms for big data frameworks* (pp. 196–217). IGI Global.
- Mei, J., Li, K., & Li, K. (2017). Customer-satisfaction-aware optimal multiserver configuration for profit maximization in cloud computing. *T-SUSC*, 2(1), 17–29.

- Mell, P., & Grance, T. (2011). The nist definition of cloud computing, computer security division, information technology laboratory, (nist).
- Midya, S., Roy, A., Majumder, K., & Phadikar, S. (2018). Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. *Journal of Network and Computer Applications*, *103*, 58–84.
- Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software*, *83*, 80–98.
- Mosa, A., & Paton, N. W. (2016). Optimizing virtual machine placement for energy and sla in clouds using utility functions. *Journal of Cloud Computing*, *5*(1), 17.
- Moschakis, I. A., & Karatza, H. D. (2011). Performance and cost evaluation of gang scheduling in a cloud computing system with job migrations and starvation handling.
- Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2017). Cloud storage reliability for big data applications: A state of the art survey. *Journal of Network and Computer Applications*, *97*, 35–47.
- Navimipour, N. J. (2015). Task scheduling in the cloud computing based on the cuckoo search algorithm. *International Journal of Modeling and Optimization*, *5*(1), 44.
- Nir, M., Matrawy, A., & St-Hilaire, M. (2018). Economic and energy considerations for resource augmentation in mobile cloud computing. *IEEE Transactions on Cloud Computing*, *6*(1), 99–113.
- Omer, S., Babiker, A., & Mustafa, A. (2011). Advantages of autonomic cloud computing comparative analysis. *Journal of Electrical and Electronics Engineering*, 56–60.
- Oprescu, A.-M., & Kielmann, T. (2010). Bag-of-tasks scheduling under budget constraints. In *Cloud computing technology and science (cloudcom), 2010 ieee second international conference on* (pp. 351–359).
- OutrightSystems. (2019). *Cloud computing in business*. Retrieved from <https://medium.com/@outrightsystems>
- Panda, S. K., & Jana, P. K. (2017). Sla-based task scheduling algorithms for heterogeneous multi-cloud environment. *The Journal of Supercomputing*, *73*(6), 2730–2762.

- Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (aina), 2010 24th ieee international conference on* (pp. 400–407).
- Perez Botero, D., Szefer, J., & Lee, R. B. (2013). Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 international workshop on security in cloud computing* (pp. 3–10).
- Poola, D., Ramamohanarao, K., & Buyya, R. (2014). Fault-tolerant workflow scheduling using spot instances on clouds. *Procedia Computer Science*, 29, 523–533.
- Prodan, R., Wiczorek, M., & Fard, H. M. (2011). Double auction-based scheduling of scientific applications in distributed grid and cloud environments. *Journal of Grid Computing*, 9(4), 531–548.
- Qiang, L. (2012). Applying stochastic integer programming to optimization of resource scheduling in cloud computing. *Journal of Networks*, 7(7), 1078.
- Qiu, X., Dai, Y., Xiang, Y., & Xing, L. (2017). Correlation modeling and resource optimization for cloud service with fault recovery. *IEEE Transactions on Cloud Computing*(1), 1–1.
- Reddy, K. S., Panwar, L. K., Kumar, R., & Panigrahi, B. (2017). Profit-based conventional resource scheduling with renewable energy penetration. *International Journal of Sustainable Energy*, 36(7), 619–636.
- Rimal, B. P., Choi, E., & Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In *Inc, ims and idc, 2009. ncm'09. fifth international joint conference on* (pp. 44–51).
- Sahni, J., & Vidyarthi, D. P. (2018). A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. *IEEE Transactions on Cloud Computing*, 6(1), 2–18.
- Salehi, M. A., & Buyya, R. (2010). Adapting market-oriented scheduling policies for cloud computing. In *International conference on algorithms and architectures for parallel processing* (pp. 351–362).
- Sampaio, A. M., & Barbosa, J. G. (2018). A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud. *Sustainable Computing: Informatics and Systems*, 19, 315–323.
- Sandholm, T., Ward, J., Balestrieri, F., & Huberman, B. A. (2015). Qos-based pricing and scheduling of batch jobs in openstack clouds. *arXiv preprint arXiv:1504.07283*.

- Saripalli, P., & Walters, B. (2010). Quirc: A quantitative impact and risk assessment framework for cloud security. In *Cloud computing (cloud), 2010 IEEE 3rd international conference on* (pp. 280–288).
- Sedaghat, M., Hernández-Rodríguez, F., & Elmroth, E. (2014). Autonomic resource allocation for cloud data centers: A peer to peer approach. In *2014 international conference on cloud and autonomic computing (iccac)* (pp. 131–140).
- Serrano, D., Bouchenak, S., Kouki, Y., de Oliveira Jr, F. A., Ledoux, T., & Lejeune. (2016). Sla guarantees for cloud services. *Future Generation Computer Systems*, *54*, 233–246.
- Sheikhalishahi, M., Grandinetti, L., Wallace, R. M., & Vazquez-Poletti, J. L. (2015). Autonomic resource contention-aware scheduling. *Software: Practice and Experience*, *45*(2), 161–175.
- Singh, S., & Chana, I. (2015). Qrsf: Qos-aware resource scheduling framework in cloud computing. *The Journal of Supercomputing*, *71*(1), 241–292.
- Singh, S., & Chana, I. (2016). Earth: Energy-aware autonomic resource scheduling in cloud computing. *Journal of Intelligent & Fuzzy Systems*, *30*(3), 1581–1600.
- Singh, S., Chana, I., & Buyya, R. (2017). Star: Sla-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*.
- Singh, S., Chana, I., Singh, M., & Buyya, R. (2016). Soccer: self-optimization of energy-efficient cloud resources. *Cluster Computing*, *19*(4), 1787–1800.
- Sivanandam, S., & Deepa, S. (2008). Genetic algorithm optimization problems. In *Introduction to genetic algorithms* (pp. 165–209). Springer.
- Son, S., & Jun, S. C. (2013). Negotiation-based flexible sla establishment with sla-driven resource allocation in cloud computing. In *Cluster, cloud and grid computing (ccgrid), 2013 13th IEEE/ACM international symposium on* (pp. 168–171).
- Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Internet computing*, *13*(5).
- Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., & Wang, J. (2013). Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing*, *39*(4-5), 177–188.
- Sun, D., Zhang, G., Wu, C., Li, K., & Zheng, W. (2017). Building a fault tolerant framework with deadline guarantee in big data stream computing environments. *Journal of Computer and System Sciences*, *89*, 4–23.

- Tafsiri, S. A., & Yousefi, S. (2018). Combinatorial double auction-based resource allocation mechanism in cloud computing market. *Journal of Systems and Software*, 137, 322–334.
- Tanenbaum, A. S. (2009). *Modern operating system*. Pearson Education, Inc.
- Tang, C., Xiao, S., Wei, X., Hao, M., & Chen, W. (2018). Energy efficient and deadline satisfied task scheduling in mobile cloud computing. In *Big data and smart computing (bigcomp), 2018 IEEE International Conference on* (pp. 198–205).
- Tao, F., Feng, Y., Zhang, L., & Liao, T. W. (2014). Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing*, 19, 264–279.
- Teng, F., & Magoules, F. (2010). Resource pricing and equilibrium allocation policy in cloud computing. In *Computer and information technology (cit), 2010 IEEE 10th International Conference on* (pp. 195–202).
- Tesfatsion, S. K., Wadbro, E., & Tordsson, J. (2016). Autonomic resource management for optimized power and performance in multi-tenant clouds. In *Autonomic computing (icac), 2016 IEEE International Conference on* (pp. 85–94).
- Thomas, E., Zaigham, M., & Ricardo, P. (2013). *Cloud computing concepts, technology & architecture*. Prentice Hall.
- Um, T. W., Lee, H., Ryu, W., & Choi, J. K. (2014). Dynamic resource allocation and scheduling for cloud-based virtual content delivery networks. *ETRI Journal*, 36(2), 197–205.
- Van, H. N., Tran, F. D., & Menaud, J.-M. (2009). Autonomic virtual resource management for service hosting platforms. In *Software engineering challenges of cloud computing, 2009. cloud'09. icse workshop on* (pp. 1–8).
- Van den Bossche, R., Vanmechelen, K., & Broeckhove, J. (2010). Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In *Cloud computing (cloud), 2010 IEEE 3rd International Conference on* (pp. 228–235).
- Vashistha, A., Kumar, S., Verma, P., & Porwal, R. (2018). A self-adaptive view on resource management in cloud data center. In *2018 8th International Conference on Cloud Computing, Data Science & Engineering (confluence)* (pp. 1–5).
- Viswanathan, H., Lee, E. K., Rodero, I., & Pompili, D. (2015). Uncertainty-aware autonomic resource provisioning for mobile cloud computing. *IEEE transactions on parallel and distributed systems*, 26(8), 2363–2372.
- Wang, L., Liu, M., & Meng, M. Q.-H. (2017). A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems. *IEEE transactions on cybernetics*, 47(2), 473–484.

- Weiwei, L., Liang, C., Wang, J. Z., & Buyya, R. (2014). Bandwidth-aware divisible task scheduling for cloud computing. *Software: Practice and Experience*, 44(2), 163–174.
- Wen, C. Y., & Chang, J. M. (2018). Fair demand response with electric vehicles for the cloud based energy management service. *IEEE Transactions on Smart Grid*, 9(1), 458–468.
- Wenbing, Z., Melliar-Smith, P., & Moser, L. E. (2010). Fault tolerance middleware for cloud computing. In *Cloud computing (cloud), 2010 ieee 3rd international conference on* (pp. 67–74).
- Wu, L., Garg, S. K., Versteeg, S., & Buyya, R. (2014). Sla-based resource provisioning for hosted software-as-a-service applications in cloud computing environments. *IEEE Transactions on services computing*, 7(3), 465–485.
- Xie, N., Zhang, X., & Zhang, J. (2017). A truthful auction-based mechanism for virtual resource allocation and pricing in clouds. In *Computer and communications (iccc), 2017 3rd ieee international conference on* (pp. 578–582).
- Xu, C.-Z., Rao, J., & Bu, X. (2012). Url: A unified reinforcement learning approach for autonomic cloud management. *Journal of Parallel and Distributed Computing*, 72(2), 95–105.
- Xuejun, L., Ding, R., Liu, X., Liu, X., Zhu, E., & Zhong, Y. (2016). A dynamic pricing reverse auction-based resource allocation mechanism in cloud workflow systems. *Scientific Programming*, 2016, 17.
- Yali, Z., Calheiros, R. N., Gange, G., Ramamohanarao, K., & Buyya, R. (2015). Sla-based resource scheduling for big data analytics as a service in cloud computing environments. In *Parallel processing (icpp), 2015 44th international conference on* (pp. 510–519).
- Yang, Z., Yin, C., & Liu, Y. (2011). A cost-based resource scheduling paradigm in cloud computing. In *Parallel and distributed computing, applications and technologies (pdcat), 2011 12th international conference on* (pp. 417–422).
- Yibin, L., Chen, M., Dai, W., & Qiu, M. (2017). Energy optimization with dynamic task scheduling mobile cloud computing. *IEEE Systems Journal*, 11(1), 96–105.
- Yuan, H., Bi, J., Tan, W., & Li, B. H. (2017). Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds. *IEEE Trans. Automation Science and Engineering*, 14(1), 337–348.
- Yusoh, Z. I. M., & Tang, M. (2012). Composite saas placement and resource optimization in cloud computing using evolutionary algorithms. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 590–597).

- Zhangjun, W., Liu, X., Ni, Z., Yuan, D., & Yang, Y. (2013). A market-oriented hierarchical scheduling strategy in cloud workflow systems. *The Journal of Supercomputing*, 63(1), 256–293.
- Zheng, Z., Zhou, T. C., Lyu, M. R., & King, I. (2010). Ftcloud: A component ranking framework for fault-tolerant cloud applications. In *2010 IEEE 21st International Symposium on Software Reliability Engineering* (pp. 398–407).
- Zhipiao, L., Wang, S., Sun, Q., Zou, H., & Yang, F. (2013). Cost-aware cloud service request scheduling for saas providers. *The Computer Journal*, 57(2), 291–301.
- Zhongjin, L., Ge, J., Yang, H., Huang, L., Hu, H., Hu, H., & Luo, B. (2016). A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. *Future Generation Computer Systems*, 65, 140–152.
- Zhou, Z., Dong, M., Ota, K., Wang, G., & Yang, L. T. (2016). Energy-efficient resource allocation for d2d communications underlying cloud-ran-based lte-a networks. *IEEE Internet of Things Journal*, 3(3), 428–438.
- Zuo, L., Shu, L., Dong, S., Chen, Y., & Yan, L. (2017). A multi-objective hybrid cloud resource scheduling method based on deadline and cost constraints. *IEEE Access*, 5, 22067–22080.

Chapters

ORIGINALITY REPORT

7 %	6 %	4 %	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijmems.in Internet Source	1 %
2	Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha, Suresh Chandra Satapathy. "Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges", Software: Practice and Experience, 2020 Publication	1 %
3	Seyedali Mirjalili. "The Ant Lion Optimizer", Advances in Engineering Software, 2015 Publication	1 %
4	link.springer.com Internet Source	1 %
5	vimcloud.org Internet Source	1 %
6	tudr.thapar.edu:8080 Internet Source	1 %
7	www.cse.dmu.ac.uk	

List of Publications

Internet Source

1%

8

file.scirp.org
Internet Source

1%

Exclude quotes Off
Exclude bibliography On

Exclude matches < 1%

CURRICULUM VITAE

BHUPESH KUMAR DEWANGAN
 PhD (Pursuing), MTech CSE (Honors), BE CSE
 C-204, Eastern Arc, Badowala, Dehradun. UK. 248171
 bhupesh.dewangan@gmail.com
 9074648648



Career Objective

To obtain a challenging job where I can put my knowledge and experience to the growth of the organization under the guidance of expert.

Educational Qualifications

I am pursuing PhD from University of Petroleum and Energy Studies. I have done Master of Technology with HONORS from Chhattisgarh Swami Vivekanand Technical University, Bilai in Computer Science and Engineering year of 2012 and have done BE from Pt. Ravishankar Shukla University, Raipur in Computer Science and Engineering year of 2005 with 74%.

S.No.	Degree	Specialization	University/ Board	Year of Passing	Grade/ Division
1	PhD	CSE	UPES, Dehradun	Enrolled in Jan 2015	NA
2	MEM.Tech./M.Sc./MCA	CSE	CSVTU, Bilai	2012	First(Honors)
3	BE/B.Sc./BCA/B.Tech.	CSE	RSSU, Raipur	2005	First
4	Diploma	Diploma in Software Testing from SEED Infotech, Pune			
5	XII	Science	MP Board	2000	First
6	X	NA	MP Board	1998	First

Certifications

- Fundamentals of Microsoft Azure, 2019.
- Diploma in Software Testing from Seed Infotech Pune. 2014
- IBM Mainframe Application from Maple ESM Technologies, Pune. 2006

Experience

Total Experience: 15+ years

Current Employer: *University of Petroleum and Energy Studies*
Designation: Assistant Professor in School of Computer Science & Engineering.
Duration: July 2017 to till date.

Previous Employer: *Chhatrapati Shivaji Institute of Technology, Durg.*
Designation: Associate Professor in CSE Department.
Duration: Oct 2007 to June 2017.

List of Publications

Previous Employer: GET (Global Education Tree) Thane.
Designation: Software Trainer. (IBM Mainframe Application).
Duration: July 2005 to Aug 2007.

Publications (PhD Work)

Summary of Publications		
1	SCI Indexed Journal	1
2	Scopus Indexed Journal	7
3	Scopus indexed Conference	2
Total		10

List of Publications (PhD Work)

Paper Published in SCI/ SCOPUS/ ESCI indexed Journals:

1. Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha. 2020. Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. *Software: Practice and Experience* 4(1). PP: 1-20. *SCI*
<https://doi.org/10.1002/spe.2810>
2. Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha. 2020. Cloud Resource Optimization System based on Time and Cost. *International Journal of Mathematical, Engineering and Management Sciences* 5(4). PP: 758-768. *SCOPUS Q2*
<http://doi.org/10.33889/IJMEMS.2020.5.4.060>
3. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Energy-Aware Autonomic Resource Scheduling Framework for Cloud. *International Journal of Mathematical, Engineering and Management Sciences* 4(1). PP: 41-55. *SCOPUS Q2*
<https://doi.org/10.33889/IJMEMS.2019.4.1-004>
4. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Design of Self-Management Aware Autonomic Resource Scheduling Scheme in Cloud. *International Journal of Computer Information Systems and Industrial Management Applications*. Vol (11), PP: 170-177. *SCOPUS Q3*
http://www.mirlabs.net/ijcisim/regular_papers_2019/IJCISIM_17.pdf
5. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha, Tanupriya Choudhury, 2019. An Automated Self-Healing Cloud Computing Framework for Resource Scheduling. *International Journal of Grid and High Performance Computing*. Accepted for Vol (13), Issue (2), Article (3). ESCI, IET Inspec. IGI Global. USA. *SCOPUS Q2*
6. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Self-characteristics based Energy-Efficient Resource Scheduling for Cloud. *Procedia Computer Science-ELSEVIER*, Vol 152. PP: 204-211. *SCOPUS*
<https://doi.org/10.1016/j.procs.2019.05.044>

7. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. Sla-Based Autonomic Cloud Resource Management framework By Antlion Optimization Algorithm. International Journal of Innovative Technology and Exploring Engineering. 8(4).119-123. *SCOPUS*. <https://www.ijitee.org/wp-content/uploads/papers/v8i4/D2659028419.pdf>
8. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2019. A Self-Optimization Based Virtual Machine Scheduling to Workloads in Cloud Computing Environment. International Journal of Engineering and Advanced Technology. 8(4). 91-96. *SCOPUS*. <https://www.ijeat.org/wp-content/uploads/papers/v8i4/D24270484C19.pdf>

Paper presented in SCOPUS indexed International Conferences:

9. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2018. Credential and Security Issues of Cloud Service Models. In proceeding of 2016 IEEE conference on Next Generation Computing Technologies Publisher IEEE *SCOPUS*. <https://ieeexplore.ieee.org/document/7877536>
<https://doi.org/10.1109/NGCT.2016.7877536>
10. Bhupesh Kumar Dewangan, Amit Agarwal, Venkatadri M, Ashutosh Pasricha. 2018. Autonomic Cloud Resource Management. Fifth IEEE international Conference on Parallel, Distributed and Grid Computing(PDGC). PP:138-143. IEEE *SCOPUS*. <https://ieeexplore.ieee.org/document/8745977>
<https://doi.org/10.1109/PDGC.2018.8745977>

Personal Profile

Name:	Bhupesh Kumar Dewangan
Father's Name:	Mr. S.R. Dewangan
Sex:	Male
Marital Status:	Married
Nationality:	Indian
Hobbies:	Working out in GYM, Singing
Phone:	9074648648
Date of Birth:	16 Sep, 1983

Declaration

I hereby declare that the above-mentioned information is correct up to my knowledge and I bear the responsibility for the correctness of the above-mentioned particulars.

Place: Dehradun (UK)


(Bhupesh Kumar Dewangan)