Energy-aware Load Balancing Techniques for Cloud Computing

*A **Thesis***

submitted for the award of degree of

DOCTOR OF PHILOSOPHY

by

Nidhi Jain

(901003001)

Under the guidance of

Dr. Inderveer Chana

Professor

Computer Science and Engineering Department

Thapar University, Patiala

Computer Science and Engineering Department

Thapar University, Patiala -147004, INDIA

May 2017

Contents

List of Figures vii
List of Tables ix
Certificate xi
Acknowledgments xii
Abstract xiv

1 Introduction **1**
1.1 Cloud Computing: An Overview 2
1.1.1 Cloud Service Models 3
1.1.2 Cloud Deployment Models 4
1.1.3 Cloud Components 5
1.2 Cloud Computing and Energy Efficiency 7
1.2.1 Cloud Data Centers 7
1.2.2 Necessitation of Energy Efficiency Measures in Cloud Data Centers 7
1.2.3 Accomplishing Energy Efficiency through Cloud Computing . . 10
1.3 Cloud Enabling Technologies 10
1.3.1 Virtualization 11
1.3.2 Load Balancing 12
1.4 Cloud Computing Challenges 14
1.4.1 Virtual Machine Migration 15
1.4.2 Server Consolidation 15
1.4.3 Energy Management in Cloud 16
1.4.4 Load Balancing 17

1.5 Research Motivation 18
1.6 Thesis Contributions 19
1.7 Thesis Organization 20

2 Literature Survey **23**
2.1 Load Balancing 24
2.1.1 Load Balancing Categories 24
2.1.2 Load Balancing Goals 26
2.1.3 Load Balancing Policies 27
2.1.4 Load Balancing Strategies 29
2.1.5 Load Balancing Metrics 29
2.2 Existing Load Balancing Techniques in Clouds 30
2.2.1 Non-energy-aware Load Balancing Techniques 31
2.2.2 Energy-aware Load Balancing Techniques 37
2.3 Existing Resource Utilization Techniques 41
2.4 Existing VM Migration Techniques 45
2.5 Bio-Inspired Techniques 52
2.6 Merits of Bio-inspired approaches for Energy Conservation 59
2.7 Problem Statement & Research Objectives 62
2.8 Conclusion 62

3 Proposed Energy-aware Resource Utilization Model **64**
3.1 Energy-aware Resource Utilization (ERU) Model 65
3.1.1 System Model 71
3.1.2 Fitness function 73
3.2 Artificial Bee Colony (ABC) Optimization Technique 74
3.3 ABC-based ERU 76
3.4 Experimental Results of ERU Model 80
3.4.1 Performance Evaluation 81
3.4.2 Discussion 84
3.5 Validation of ERU with Existing Technique 85

3.5.1 Discussion 87
3.6 Conclusion 88

4 Proposed Energy-aware VM Migration Technique 89
4.1 Firefly Optimization (FFO) Algorithm 90
4.2 Proposed FireFly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) Technique 91
4.3 Experimental Results of FFO-EVMM Technique 97
4.3.1 Performance Evaluation 97
4.3.2 Discussion 101
4.4 Validation of FFO-EVMM with Existing Technique 104
4.4.1 Discussion 106
4.5 Conclusion 107

5 Proposed Energy-aware Load Balancing Techniques 108
5.1 Proposed Energy-aware Load Balancing (ELB) Model 109
5.1.1 Modules of ELB Model 111
5.1.2 Phases of ELB Model 113
5.2 Energy-aware Load Balancing Technique with Resource Utilization - ELB(RU) 115
5.3 Energy-aware Load Balancing Technique without Resource Utilization - ELB(w/o RU) 120
5.4 Experimental Results of ELB Techniques 122
5.4.1 Performance Evaluation 122
5.4.2 Discussion 125
5.5 Validation of Proposed ELB Techniques 125
5.6 Conclusion 128

6 Case Study : Empirical Evaluation of Load Balancing Techniques at BSNL Data Center 129
6.1 About BSNL Data Center 130

6.2 Experimental Results 132
6.3 Experimental Results of ELB(RU) technique 134
6.3.1 Test Case 1: Resource utilization (CPU & Memory) 135
6.3.2 Test Case 2: VM Migrations 141
6.3.3 Test Case 3: Nodes Used 142
6.3.4 Test Case 4: Energy Consumption 143
6.3.5 Test Case 5: Load Balancing 146
6.3.6 Test Case 6: Performance 151
6.3.7 Discussion 152
6.4 Experimental Results of ELB(w/o RU) Technique 154
6.4.1 Test Case 1: Performance 155
6.4.2 Test Case 2: VM Migrations 156
6.4.3 Test Case 3: Nodes Used 157
6.4.4 Test Case 4: Load Balancing 158
6.4.5 Test Case 5: Energy Consumption 159
6.4.6 Discussion 160
6.5 Statistical Analysis of Results 161
6.6 Conclusion 163

7 Conclusions and Future Directions **164**
7.1 Conclusions 165
7.2 Future Research Directions 167

Bibliography **169**

List of Papers Published **190**

List of Figures

1.1 Three Service Models of Cloud Computing 4
1.2 Four Deployment Models of Cloud Computing 6
1.3 Percentage contribution of various industrial divisions towards GHG emissions throughout the globe [33]. 8
1.4 Server with Virtualization 11
1.5 Classification of Load balancing 13
1.6 Green Computing in Clouds 17

2.1 Load Balancing Taxonomy 25

3.1 Energy-aware Resource Utilization Model 66
3.2 Flowchart for Energy-aware Resource Utilization Model 69
3.3 Total ET: ERU vs FFD vs ACO 83
3.4 CE : ERU vs FFD vs ACO 84
3.5 EC : ERU vs FFD vs ACO 84
3.6 Improvement Results of ERU over Other Techniques 85
3.7 VMs vs Nodes 86
3.8 VMs vs Energy Consumption 87
3.9 Improvement Graph of ERU over ESWCT 87

4.1 Flowchart for FFO-EVMM Technique 96
4.2 VMs vs Nodes 99
4.3 VMs vs Number of Migrations 99
4.4 VMs vs Energy Consumption 100

4.5 Thresholds vs Avg Energy Consumption 101
4.6 Overall improvement of FFO-EVMM 101
4.7 Improvement Graph of FFO-EVMM over FFD 102
4.8 Improvement Graph of FFO-EVMM over ACO 102
4.9 Percent Improvement of FFO-EVMM over Other Techniques 103
4.10 Performance Analysis 103
4.11 VMs vs Nodes 105
4.12 VMs vs VM Migrations 105
4.13 VMs vs Energy Consumption 106
4.14 Improvement Graph of FFO-EVMM & MPSO 107

5.1 Energy-aware Load Balancing Model 110
5.2 Flowchart for ELB(RU) Technique 119
5.3 Flowchart for ELB(w/o RU) Technique 121
5.4 VMs vs VM Migrations 123
5.5 VMs vs Energy Consumption 124
5.6 Improvement Graph of ELB over EnhancedLB 125

6.1 BSNL Data Center 130
6.2 Servers in BSNL Data Center 131
6.3 Execution of the Load Balancing Process 135
6.4 Percentage of CPU Utilization by each Technique 136
6.5 CPU Utilization of RR 136
6.6 CPU Utilization of FFD 137
6.7 CPU Utilization of ACO 137
6.8 CPU Utilization of ELB(RU) 138
6.9 Percentage of Memory Utilization by each Technique 138
6.10 Memory Utilization of RR 139
6.11 Memory Utilization of FFD 139
6.12 Memory Utilization of ACO 140
6.13 Memory Utilization of ELB(RU) 140

6.14 VM Migrations Performed by each Technique 141
6.15 Nodes Used by each Technique 142
6.16 Energy Consumed by each Technique 144
6.17 Energy Consumption of RR 144
6.18 Energy Consumption of FFD 145
6.19 Energy Consumption of ACO 145
6.20 Energy Consumption of ELB(RU) 146
6.21 Servers Used by each Technique before Migration 147
6.22 Servers Used by each Technique after Migration 148
6.23 Servers Used by RR 148
6.24 Servers Used by RR 149
6.25 Servers Used by FFD 149
6.26 Servers Used by ACO 150
6.27 Servers Used by ELB(RU) 150
6.28 Servers Used by ELB(RU) 151
6.29 Performance of each Technique in terms of Handled Workloads 151
6.30 Percentage Improvement of FFD, ACO & ELB(RU) over RR 152
6.31 Percentage Improvement of ELB(RU) over RR, FFD & ACO 153
6.32 Overall Enhancement Graph of ELB(RU) 153
6.33 Number of Workloads Handled by each Technique 155
6.34 Number of VM Migrations Performed by each Technique 156
6.35 Number of Nodes used by each Technique 157
6.36 Servers Used by each Technique before VM Migrations 158
6.37 Servers used by each Technique after VM Migrations 159
6.38 Energy Consumption by each Technique 160
6.39 Percentage Improvement of ELB(w/o RU) Technique over Others . . . 160
6.40 Coefficient of Variation for the Energy Consumption with each Technique 162
6.41 Coefficient of Variation for the Resource Utilization with each Technique 162
6.42 Coefficient of Variation for the Performance with each Technique 163

List of Tables

2.1 Analysis Table of Existing Non-energy-aware Load Balancing Techniques 34
2.2 Analysis Table of Existing Energy-aware Load Balancing Techniques . . 39
2.3 Analysis Table of Existing Resource Utilization Techniques 43
2.4 Analysis Table of Existing Virtual Machine Migration Techniques . . . 49
2.5 Comparison of Different Bio-inspired Approaches 58
2.6 Comparison of Bio-inspired Computing-based Approaches [33] 60

3.1 Description of Various Terms 66
3.2 Initialization Parameters 77
3.3 Simulation Parameters 81
3.4 Node Description 82
3.5 VM Description 82

4.1 Example 95
4.2 Simulation Parameters 97
4.3 Node Description 98
4.4 VM Description 98

5.1 Node Description 122
5.2 VM Description 122
5.3 Validation Table of Proposed ELB Techniques with Existing Techniques 126

6.1 System Configuration 131
6.2 Abbreviations Used 132
6.3 Server Description 132

6.4 Software Configuration of each server 133

Dedicated to

my Family

Certificate

I hereby certify that the work which is being presented in this thesis titled, **"Energy-aware Load Balancing Techniques for Cloud Computing"**, in the fulfillment of the requirement for the award of degree of **"Doctor of Philosophy"** submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Inderveer Chana and refers other researchers' works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Nidhi Jain

(Nidhi Jain)
901003001

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

Inderveer 17/05/17

(Inderveer Chana)
Professor
Computer Science and Engineering Department
Thapar University
Patiala

Acknowledgements

At the outset, I express my gratitude to the Divine Force, Who was like a lamp to my feet and filled me with the zeal,enthusiasm, wisdom and perseverance to complete the challenging journey of this PhD thesis successfully. I am eternally grateful to Him for his continuum of blessings.

I would like to express my heartfelt gratitude to my worthy supervisor, Dr. Inderveer Chana, Professor, Thapar University, Patiala, for being a beacon of light and continuously guiding me through thick and thin. She indeed is a treasure trove of all the qualities that a pupil could have asked for in a mentor. Her constant support, encouragement, immense knowledge and scientic perception, lit up my way in the darkest times. I am eternally indebted to her for her insightful discussions and path breaking suggestions that assisted me in shaping up the direction of this research. The time and energy that she devoted on careful and the patient proof readings of this manuscript helped me to present this thesis with the desired quality. I cannot imagine a better mentor for my PhD study. Her belief in me was always re-energizing and rejuvenating.

I extend my sincere thanks to the members of my Doctoral committee - Dr. Seema Bawa, Dr. V.P.Singh and Dr. Kulbir Singh for their academic support and invaluable comments. I sincerely concede their valuable feedback and constructive comments while ensuring the progress of my research work. I will forever be thankful to Dr. Maninder Singh, Head, Computer Science & Engineering Department, Thapar University, Patiala, for his whole hearted support & for providing productive and stimulating ideas that motivated me to explore the tools and technologies for carrying out the experiments during this research process. My deepest regards and gratitude to Dr. O.P.Pandey, Dean of Research & Sponsored Projects and Dr. Prakash Gopalan, Director, Thapar University for all the facilities that have been instrumental in the creation of a healthy research environment in the university and have been immensely helpful for the completion of my work.

I wish to further extend my thanks to all the members of the faculty of the Computer Science & Engineering Department, Thapar University, Patiala, who helped me in one way or the other to carry out my research work successfully. I am grateful to them for their co-operation and support. I also acknowledge the cooperation rendered to me by the office and the laboratory staff of this department. I would also like to thank all

my friends and lab mates for their continuous motivation and moral support. Also, I am thankful to them for all the great times that we have shared and for making this research experience enjoyable and memorable.

I gratefully acknowledge the generous assistance provided by GSM data center team of BSNL, Chandigarh, India. I also express my appreciation to the organization for granting me an opportunity to work on its infrastructure.

This thesis would never have been conceived or borne fruit without the unconditional support of my family. My parents deserve a special mention here, who inculcated moral, ethical and religious values in me. They made me smile and realize their faith in me during the rough road of this journey. They have passed onto me a wonderful humanitarian lineage and a good foundation to face the challenges of life. I gratefully acknowledge the patience and love of my brother and his family that helped me in every sphere of my life.

I would extend my deep sense of gratitude to one and all in my parents-in-law family who extended their cooperation and encouragement to me. My in-laws family provided me an excellent conducive environment for continuing my research.

Last but not the least, my warmest thanks to my dear husband, whose unconditional support and continuous motivation during all these years is beyond expression in words. His was the supporting hand that guided the first baby steps and his are the encouraging words that made sure that I concluded what I began. He inspired me in all the dimensions of life and instilled condence in me to successfully complete this journey. I will always be indebted to him for his everlasting love and commitment that carries me through always. I look forward for a life full of happiness and togetherness.

These acknowledgements would remain incomplete if I do not mention my kids, who patiently waited for the completion of their Moms work. My kids have been my pillars of strength. I owe everything to them and this thesis would not have been completed without their everlasting love and support.

Nidhi Jain

Abstract

Cloud computing, a form of distributed computing, promises to deliver reliable services through next-generation data centers, built on virtualized compute and storage technologies. Cloud providers rely on large data centers to offer resources required by users but the energy consumed by cloud infrastructures has lately become a key environment and economic concern. Lot of energy is wasted in these data centers due to the under-utilized resources and therefore these under-utilized resources should be efficiently utilized to conserve energy.

Another important concern in cloud computing is load balancing. Load Balancing is essential for distributing dynamic workloads over multiple nodes so that it is ensured that no single node is overwhelmed. It thus helps in avoiding hot-spots and enhancing resource utility levels thereby, reducing energy consumption. As energy efficiency has appeared as the utmost essential design requirement for current computing systems, it determines the need of energy-aware load balancing in clouds to overcome the issue of energy-efficiency.

This research work proposes two Energy-aware Load Balancing (ELB) techniques. These techniques are based on the proposed Energy-aware Resource Utilization (ERU) model and the Firefly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) technique.

The proposed ERU model efficiently manages cloud resources and enhances their utilization by allocating the jobs to the appropriate resources, using the Artificial Bee Colony (ABC) optimization approach. It also maximizes the energy-efficiency of the cloud data centers through optimal resource usage, without degrading the system performance.

Thereafter, the FFO-EVMM technique has been proposed that migrates the maximally-loaded VM to the least energy-consuming node, to reduce the consumed energy in the cloud data centers at run-time. The proposed technique intends to enhance the energy-efficiency through optimum migration of VMs, thereby improving resource utilization levels.

Finally, the ELB model has been proposed that helps the proposed load balancing techniques to take energy-aware decisions, from the perspectives of, both the provider and the user. The two ELB techniques, namely ELB(RU) and ELB(w/o RU), aid the

system administrator to balance system's load, across the minimum required active nodes, in an energy-aware manner. The ELB(RU) technique uses an amalgamation of ERU & FFO-EVMM. Through ABC-based ERU, the ELB(RU) technique aspires to maximize resource utilization by appropriately assigning the users workloads to the energy-aware nodes. Furthermore, by using the FFO-EVMM technique, it migrates the most loaded VMs to the least energy-consuming nodes, thereby curtailing the energy needs of the cloud data centers. The ELB(w/o RU) technique strives to augment the performance of the system by processing maximum number of workloads. It uses only the FFO-EVMM algorithm for energy-aware VM migrations, consequently avoiding hot-spots and bringing down the energy consumption levels in the cloud data centers.

The performance of the proposed ABC-based ERU model and the FFO-EVMM technique has been evaluated by comparing them with the First Fit Decreasing (FFD) and the Ant Colony Optimization (ACO) algorithms, through CloudSim toolkit. The results demonstrate that an average of 8.68% of nodes and 8.66% of energy have been conserved using ERU over FFD and ACO-based techniques. Whereas, an enhancement in the average energy consumption of about 44.39% has been attained by reducing an average of 72.34% of migrations and saving 34.36% of hosts, thereby, making the data center more energy-aware.

A case study has been conducted at the data center of a telecom service provider "Bharat Sanchar Nigam Limited (BSNL), Chandigarh, India", to test and validate the proposed ELB techniques. The competence of the proposed ELB(RU) technique is exhibited by comparing it with the Round Robin (RR) technique which is currently being used in BSNL data center and also by comparing it to FFD & ACO. The adequacy of the proposed ELB(RU) technique is accomplished by saving 40.47% of the average energy consumption, enhancing 49.68% of CPU utilization level, 24.41% of memory utilization level, by lowering 63.10% of VM migrations and reducing 53.21% of nodes. Whereas, the effectiveness of the ELB(w/o RU) technique is presented by comparing it with RR, FFD, ACO & ELB(RU). The results demonstrate that ERU(w/o RU) enhances the average performance by 38.55%. It further attains 20.53% of drop in the average energy consumption by reducing 45.57% of average VM migrations and by saving 19.64% of nodes on an average. The improved results illustrate that the proposed ELB techniques effectively balance the load, enhance resource utilization and performance levels, thereby curtailing the levels of energy consumption in the cloud data centers.

The empirical evaluation clearly justifies the role of ELB(RU) to be more useful for aggrandizing the resource utilization levels and ELB(w/o RU) to be more effective in

augmenting the performance of the system. As per the Cloud user or Cloud provider requirement, the appropriate algorithm can be chosen.

The obtained results have been verified statistically by using the Coefficient of Variance (CoV), to approve the stability of the proposed ELB techniques.

By reducing consumed energy, all the proposed techniques, indirectly reduce carbon emissions and cooling requirements of the cloud data centers, leading to a further drop in the energy demand and helping in achieving green computing.

Chapter 1

Introduction

Distributed computing integrates geographically scattered computers and distributes their workload across each computer in such a way so as provide reliable, compatible and extensive access to the high-end computation. Different paradigms of distributed computing apparently cluster, grid and cloud computing have appeared over the years.

Cloud computing, has emerged as a new paradigm and the latest evolution of distributed computing that relocates processing and data from desktops and portable personal computers, into huge data centers. To rapidly and effortlessly adjust the capacity of the data centers, the scalable Information Technology (IT) resources as well as the underlying infrastructure are offered on pay-per-use basis over the Internet, thereby, accommodating the variability in demand and helping any organization to avoid the extra financial burden. Apart from this, it helps to efficiently manage energy wastage problems. The dilemma of energy crisis actuates the indispensability of cloud computing, as it is observed to be the most preferable paradigm for realizing energy efficiency.

This chapter provides a detailed level view of the thesis. It discusses the fundamental concepts related to the cloud computing along with the importance and necessity of achieving energy efficiency. It further unfolds its close alliance with other underpinning technologies and provides the major issues of this area. Henceforth, it provides the motivation of the research and culminates with the discernment of the contributions and the organization of the rest of the thesis.

1.1 Cloud Computing: An Overview

Cloud computing, has emerged as a new paradigm of distributed computing that has migrated the computing and the data from desktops, into huge data centers [1]. It is an internet-based approach for empowering suitable, on-demand network access to a communal pool of computing resources like networks, servers, storage, applications, and services etc. These resources can be provisioned quickly and de-provisioned with nominal management or interaction of service provider, which in turn stimulates accessibility [2] [3] [4] [5] [6]. To rapidly and effortlessly adjust the capacity of data centers, the scalable IT resources as well as the underlying infrastructure are offered on pay-per-use basis over the Internet, thereby, accommodating the changes in demand and helping any organization to avoid the capital costs of software and hardware [7] [8] [9]. The National Institute of Standards and Technology has defined cloud computing as "a model that facilitates expedient and dynamic access to a large pool of computing resources that can be shared, dynamically allocated, and discharged without much managerial involvements or service provider assistance" [10] [11] [12].

Cloud computing architecture establishes itself on certain key characteristics, a few of which are overlapping with Grid, Utility, Cluster and distributed computing [13] [5] [6]. These characteristics are enlisted as follows:

Multi-tenancy - The services owned by multiple cloud providers in a cloud environment are co-located in a single data center, which is called multi-tenancy. The layered architecture of cloud computing paradigm naturally divides the responsibilities, where each layer focuses only on their respective objectives of that layer. Multi-tenancy along with its benefits, also brings the difficulties in managing and coordinating interactions among various involved stake-holders.

Shared Resource Pooling - The cloud infrastructure provider generally offers a big pool of computing resources such that they can dynamically be assigned and re-assigned to the multiple cloud service consumers. This capability of dynamically assigning the resources enhances flexibility in managing the resource utilities and operating costs of the cloud service providers.

Geographic Distribution & Ubiquitous Network Access - The services of the cloud are accessible through Internet and hence use the Internet as a network for delivering the services to cloud users. Moreover, in order to offer improved network performance, most of the cloud data centers are distributed geographically. Thereby cloud vendors can benefit from this geo-diverseness and thus realize higher utility of services.

Service Oriented - The cloud paradigm employs a service-driven operating model and therefore strongly emphasizes on the service management. Each service vendor offers IaaS, PaaS & SaaS cloud services in accordance with the negotiated Service Level Agreement (SLA) with the clients. Thus, every cloud service provider ensures integrity and service provisioning with SLA assurance.

Dynamic Resource Provisioning - This is a key characteristic of cloud computing, where the computational capability can be acquired and released dynamically. Contrarily, the conventional computing involved resource provisioning as per the extremum demand. The resource allocation carried out dynamically facilitates service deliverers in obtaining the resources as per the required demand. This aids in lowering the operating costs significantly.

Self Organizing - The resource allocation and de-allocation is done on demand basis, empowering the service providers to manage their resource utilization as per their needs. The programmed resource management feature enables the service vendors to quickly react to the prompt changes in the service demand as surge computing.

Utility Based Pricing - A significant characteristic of cloud is that it offers a pay-per-use pricing model. Although, the pricing schemes vary according to the requested services. The utility pricing models considerably lower the operating costs because the users are charged on a per-use basis. This feature of cloud computing introduces complexities in controlling the operating cost.

After giving the overview and the key characteristics of cloud computing, the following subsections deal with various service models, deployment models and the components of cloud computing respectively.

1.1.1 Cloud Service Models

Cloud computing services are classified into three classes as shown in Figure 1.1.The three types of cloud service models offered by cloud computing are described below [5] [6] [14]:

Software-as-a-Service (SaaS) - It is a multi-tenant platform for offering applications on the Internet, boxed as a distinguished service for users to access, for example Google Docs, Face book etc.

Platform-as-a-Service (PaaS) - It is a framework that provides a specific computing platform where applications and services are developed, i.e. it is a model that offers building, testing, deployment, and hosting environments for applications created by users. e.g. Microsoft Azure and Google App Engine.

Infrastructure-as-a-Service (IaaS) - It is a framework that provides entire computing resources through a service. Users can hire or purchase required computing resources for use without operating or managing infrastructure [15]. Examples include Amazon EC2, Eucalyptus, and Nimbus.

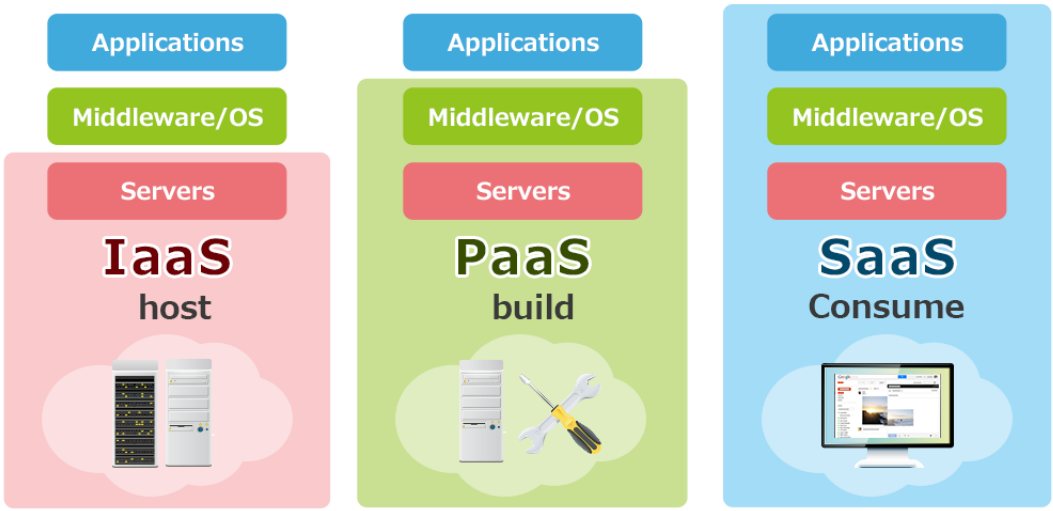


Figure 1.1: Three Service Models of Cloud Computing

The researchers have further classified the cloud services into Hardware-as-a-Service (HaaS) and Data-as-a-Service (DaaS) [16] [17] [18].

1.1.2 Cloud Deployment Models

Cloud computing is mainly classified (as shown in the Figure 1.2) on the basis of variation in the physical location and distribution, i.e. the cloud can be implemented through various types of deployment models as described below [5] [6]:

Public Cloud - Public cloud can be defined as "a cloud that is made available in a pay-as-you-go manner to the general public" [1]. i.e. these clouds are accessed by the common people and are maintained by cloud service providers like Microsoft, Google, Amazon etc. by charging the users according to their usage. These are applicable when service provider wants users to access all the resources over a network.

Private Cloud - Private cloud can be defined as an "internal data center of a business or other organization, not made available to the general public" [1], i.e. a type of proprietary computing architecture which is exclusively devised for a single organization and can not be accessed or shared with other organizations. In contrast to the public clouds, the private clouds are more expensive and secured. Private clouds are suitable when an organization wants the data to be private and/or controlled, for example, eBay and HP CloudStart.

Community Cloud - Community cloud is a type of cloud that is shared among various organizations and supports a specific community with a common tie. This type of cloud can be made available on or off premises and is managed by a mediator providing the cloud service [19] [6].

Hybrid Cloud - It is an augmentation of a private cloud with the computing capacity from public clouds [20], i.e. the cloud environment in which several internal or external service providers provide services to many organizations. In hybrid clouds, an organization can use both the private and the public clouds together. Such type of cloud is used in situations like "cloud bursting" which is the methodology for temporal leasing of capacity to manage spikes in the load [21]. For an efficient delivery of services, big IT giants such as HP, IBM, Oracle, and VMware are striving to develop and benefit from hybrid domain [16] [15].

1.1.3 Cloud Components

Cloud computing constitutes a virtualized pool of infrastructure resources with applications and services that can be used directly through a self-service portal. Cloud computing architecture consists of the following components [5] [6]:

Clients - A client interfaces with a cloud through a pre-assigned, subtle layer of abstraction. This layer carries out communication between the user requests and the displayed data which is returned in a simple and intuitive way for the user, e.g. Web Browser or a thin client application.

Data Centers - It is a collection of the different type of servers where the application is deployed and is available to various clients upon subscribing. It is a repository, either virtual or physical, used for the storage, management and distribution of the data as well the information organized in the context of a particular topic.

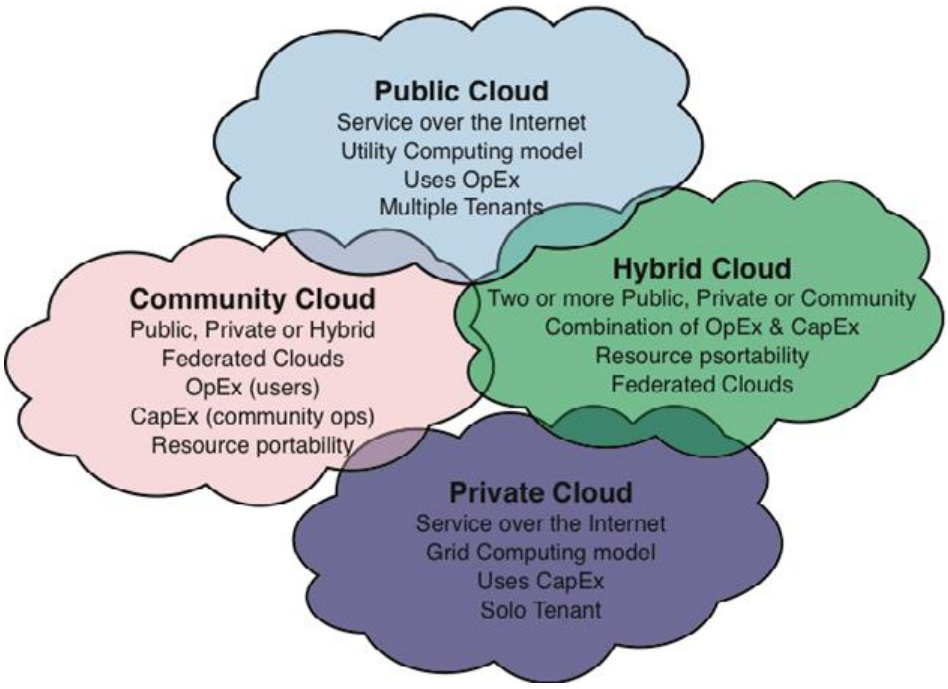


Figure 1.2: Four Deployment Models of Cloud Computing

Distributed Servers - These are the servers which are present in geographically distributed locations, thereby, enabling the service providers more flexibility in terms of operations as well as security. The implementation of the distributed servers requires various techniques for maintaining the central control as well as the coordination of the information related to the configuration of the involved servers.

Cloud Network - A network is the link connecting the wide user-base with the available cloud service providers. Undoubtedly, the Internet is the most straight-forward and popular choice to act as the cloud network. Although, advanced network services, such as compression, encryption, decryption during data at rest and data at transit will be beneficial for both the cloud user as well as the cloud service provider.

Cloud API - A cloud Application Programming Interface (API) contains a set of programmable instructions and techniques that provides an abstraction over a cloud provider. It encompasses a customized or a discrete provider call which may be used to increase the amount of control over an implementation of cloud. API calls are used to build applications for communicating with and accessing the cloud services.

After reviewing the fundamentals of cloud computing such as its key characteristics, service & deployment models and its components, the next section unfolds its role in

achieving energy efficiency.

1.2 Cloud Computing and Energy Efficiency

Cloud computing is growing exponentially resulting in a swift enlargement of data-centers thereby triggering the energy use and posing a concern of energy-efficiency improvement in data centers. This section describes the importance and necessity of achieving energy efficiency by citing the impact of a cloud in minimizing energy demand.

1.2.1 Cloud Data Centers

Cloud computing proposes an intensely networked, scalable, and virtualized environment to the cloud users without owning and managing the systems [22] [23]. Instead, the cloud users bank on the enormous space and processing capacity delivered by various service vendors [24]. Numerous data centers established to provide variable, nimble and effective services, empower the clients by immediately accessing the services available through the creation of virtual instances. The data and services are accessible perpetually without being location dependent [25]. A wide variety of IT companies, namely Google, Amazon, Salesforce, Yahoo, Microsoft, and Facebook provide pay-as-you-go cloud services by establishing their own data centers [15] [26].

1.2.2 Necessitation of Energy Efficiency Measures in Cloud Data Centers

Since the advancements in computing industry, there has been a growth of high energy consumption problems and consequently a high energy demand has been noticed. This section exhibits the consequences of increasing energy consumption in data centers and measures to achieve energy efficiency in them.

1.2.2.1 Energy Consumption Augmentation in Data Centers

With the growth and expansion of Information and Communication Technology (ICT) sector, infrastructural requirements changed and gained specialization [27]. Subsequently, Internet Data Centers (IDCs) were constructed to centralize and implement computation within specialized centers, which resulted in a sudden rise in electricity

consumption levels [28] [29] [30]. As a consequence, many peer-reviewed studies were conducted to analyze the problems and levels of energy consumption in the data centers. On the basis of the analysis offered in different reports, the ICT industry was considered to be the major energy consumer and the increase in energy consumption became a problem. The International Energy Outlook has surveyed and predicted, "The energy consumption throughout the world will grow by 56 percent between 2010 and 2040, and ICT industry will be the major culprit" [31].

1.2.2.2 Growing Energy Consumption Levels and its Impingements

Increasing energy consumption levels are complemented with alarmingly high-carbon and Green House Gas (GHG) emissions (million tonnes) into the environment, posing a danger to the environmental endurance. Figure 1.3 shows the leading perpetrators belonging to variable industrial sectors throughout the globe that subsidize to the amount and percentage level of GHG emissions [32].

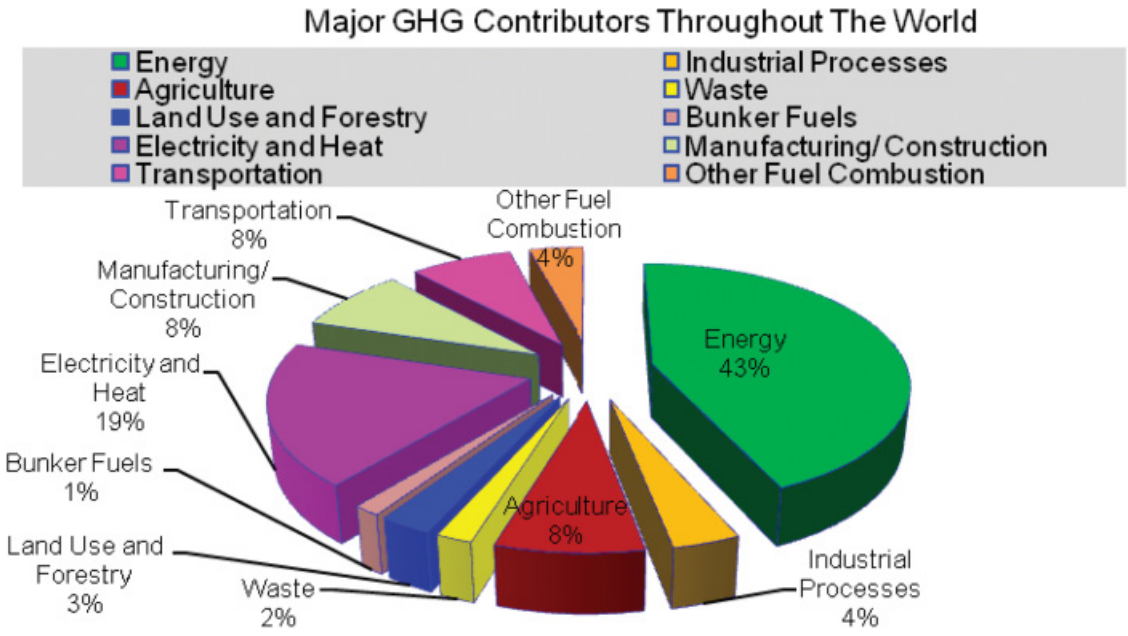


Figure 1.3: Percentage contribution of various industrial divisions towards GHG emissions throughout the globe [33].

The energy sector is clearly the major GHG emitter throughout the world. According to a report by Smart2020 [34], the overall emissions from all sources will increase by 30 percent, while the emissions by the ICT sector will reach up to 180 percent from

2002 to 2020 [35]. The rising energy consumption increases the Total Cost of Acquisition (TCA) and the Total Cost of Operation (TCO) [36], further affecting the economic budget of the service providers [37]. Even, Google needs to implement both the real and the virtual results in its data centers to decrease the operating expenses [38] [39]. The energy-saving efforts such as energy efficiency techniques are the need of the hour to meet the increasing energy demands and to reduce costs [40].

1.2.2.3 Measures for Realizing Data Center Energy Efficiency

Several endeavors have been undertaken to promote energy consciousness in ICT data centers. The attempts were made to manage power distribution among the equipment to prevent undesired energy losses and power wastage. The use of energy-efficient equipment within the data centers was also emphasized [41] [28]. The construction of green buildings and placement of high-power servers were emphasized to minimize the consumed energy in the data centers [42]. The major ICT giants such as Microsoft, IBM, and Google recently made energy optimization efforts by establishing their data centers near locations where inexpensive hydroelectric energy is readily available with complementary weather conditions [38].

However, despite all these measures, the energy consumption levels did not drop upto the desired levels. Consequently, the focus shifted onto implementing various hardware-based energy optimizations, such as using energy-efficient servers etc., the circuit-level optimizations, the logic-level optimizations and the architecture-level optimizations. However, energy efficiency did not mount despite using ideal hardware and hardware-based energy efficiency approaches. This was due to the use of poor software designs and programs [41]. As a result, the attention was switched to the energy-efficient software techniques [43]. Later on, this was observed that the energy consumption also depended on the resource management strategies, therefore a trend of developing energy-efficient algorithms to effectively provision resources while accommodating available workload and performance requirements was initiated [41].

Even though the technological platforms gradually developed, the attempts to curtail the energy consumption endured at different optimization levels. The energy-efficient mechanisms were continued to be implemented in several domains. But, cloud computing acquired captivation when there was an increase in the server utilization levels and efficient infrastructural organization as a result of cloud implementations. Consequently, the lower energy demands have led to the development of the energy-efficient techniques through cloud computing [27]. The variable number of energy

optimizations in cloud computing at different levels proved beneficial in minimizing energy consumption.

1.2.3 Accomplishing Energy Efficiency through Cloud Computing

The issue of energy crisis actuates indispensability to seek and eradicate the reasons for mounting energy consumption. The energy consumption levels are significantly influenced with the growing number of ICT equipment and its use [44]. Moreover, the growing levels of energy consumption need to be managed due to the shrinking levels of the nonrenewable energy sources [45].

Cloud computing has been observed as a solution that is endowed with the competence to be more energy efficient in comparison to the foregoing computing paradigms [46]. It offers energy efficient solutions due to the cloud enabling technologies. In addition to providing on-demand, broader network access, and rapid elastic services, it assists in managing the energy wastage problems efficiently [47] [48] [49]. It has been anticipated to be the most favored technology to prevail over the energy crisis and to realize energy efficiency. It has been prophesied that cloud computing can be an energy saving tool [27] [50] [51].

This section discussed about cloud data centers, impact of growing energy consumption levels and energy efficiency measures in the data centers followed by selecting cloud computing for realizing energy efficiency. The next section unfolds its close alliance with the cloud enabling technologies.

1.3 Cloud Enabling Technologies

Cloud computing is actualized with the help of various technologies. The most basic technology is virtualization as it empowers maximum utilization of underlying resources, followed by load balancing, that helps in distributing overall workload equally all over the cloud nodes. Finally, the need of the hour is to achieve green computing which improves the efficiency of computing resources to reduce energy consumption in cloud data centers.

1.3.1 Virtualization

Virtualization is, creating a virtual version of the hardware using the software. It enables the system to execute multiple applications simultaneously on a single physical server that run on VMs hosted on the server [52]. The simultaneous running of multiple VMs on a physical server efficiently improves the utility, which is one of the primary objectives of cloud paradigm [53].

The VMs are hosted and managed by the virtualization layer. It is the software responsible for hosting and managing all the VMs. It is a hypervisor running directly on the hardware. Hypervisor plays an important role in the virtualization scenario by virtualization of hardware. It provides support for running multiple operating systems concurrently in virtual servers created within a physical server [54]. A few popular hypervisors are: VMWare, Xen, KVM.

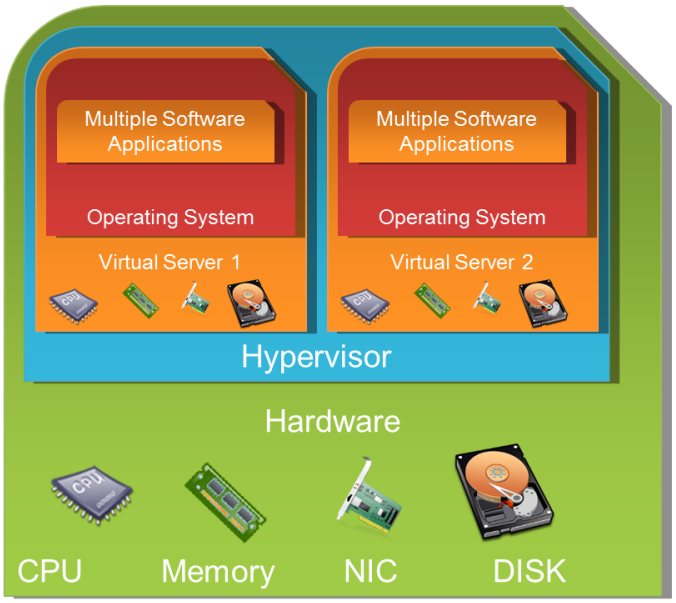


Figure 1.4: Server with Virtualization

As shown in Figure 1.4, in contrast to a server without virtualization, a server with virtualization can run multiple OS simultaneously, where each OS can have different hardware configurations, thereby, efficiently utilizing the hardware resources. As can be seen clearly from the Figure, each virtual machine is independent of the other. Virtualization thus, saves electricity, initial cost to buy servers, space etc. Also, it is easy to manage and monitor the virtual machines centrally with the help of hypervisors [55].

Hypervisors are implemented in the following two approaches:

Type I Hypervisor - Also called Bare metal Approach. It runs the hypervisor directly on the system hardware. This approach may require hardware assisted virtualization technology supported by the CPU. But, there are only a limited set of hardware drivers provided by the hypervisor vendor. Examples of the type I hypervisors are: Xen, VMWare, ESXi.

Type II Hypervisor - Also called Hosted Approach. In this implementation of the hypervisor, it runs virtual machines on top of a host OS (windows, Unix etc.). Thus, it relies completely on the host OS for physical resource management. The host operating system provides drivers for communicating with the server hardware. The most popular type II hypervisor is VirtualBox.

Hypervisor Virtualization can further be divided into following types:

Full virtualization - It enables hypervisors to run an unmodified guest operating system (e.g. Windows 2003 or XP). The guest operating system is unaware about being virtualized. e.g. A blend of direct execution and binary translation techniques, has been used by VMware to successfully realize the fully virtualized server systems.

Para virtualization - It involves explicitly modifying hooked up OS, so that it is aware of being virtualized to allow near native performance. This technique improves performance and lowers the overhead. e.g. Xen supports both Hardware Assisted Virtualization (HVM) and Para-Virtualization (PV).

Paravirtualization comprises of the modification of the operating system kernel to substitute non-virtualized instructions with hypercalls. These calls directly interact with the hypervisor that facilitates hypercall interfaces for other perilous kernel processes. On the contrary, in full virtualization, the unmodified operating system is oblivious of being virtualized and binary translation trap the delicate OS calls.

Hardware-assisted virtualization - It is also known as accelerated virtualization, hardware virtual machine (Xen) or native virtualization (Virtual iron). Under this virtual environment, confidential and subtle calls automatically entrap the hypervisor, removing the need for binary translation as conducted in full or para-virtualization.

1.3.2 Load Balancing

The coupled resources operate separately or in collaboration with each other in case of distributed environment. All these coupled resources have some initial workloads

assigned to them individually and can have distinct processing capacity. This workload is uniformly assigned to all the above resources based on their computational speed to diminish the time required to execute all the workloads [56] and thus the need of the load balancing arises.

Load balancing is the technique in which the tasks are dispensed uniformly to each and every processor, so that the state of some nodes being over-laden while others being lightly-laden or even idle can be avoided. Thereby, ensuring that the available resources are utilized maximally thus improving the response time of applications submitted by

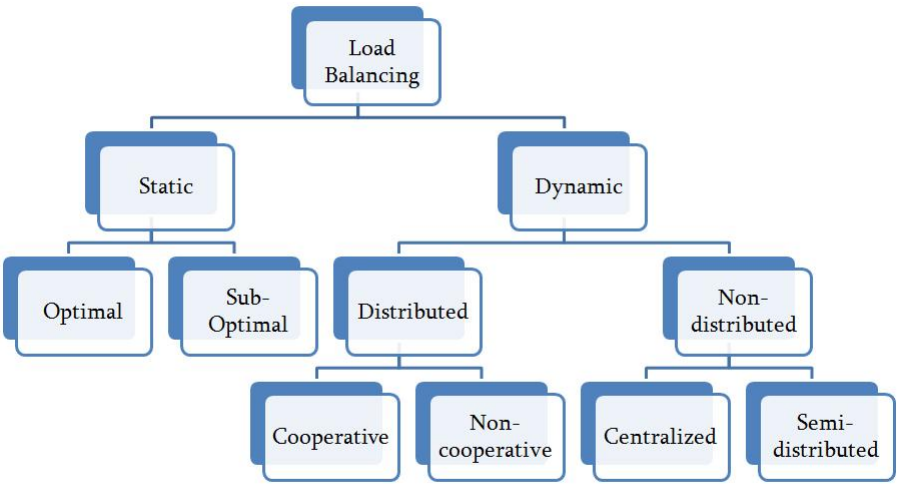


Figure 1.5: Classification of Load balancing

the user [57] [58] [59]. It can be classified into the following categories as shown in Figure 1.5.

Load balancing techniques can be classified as the Static Load Balancing (SLB) and the Dynamic Load Balancing (DLB) techniques. In SLB, the assignment of the tasks to processors is done deterministically or probabilistically at the compile time using a prior knowledge about the tasks and the system [60] [61], whereas in DLB, load distribution decision is made at the run-time by making use of the current processes and the system state information [60]. SLB can further be classified into Optimal SLB & Sub-Optimal SLB [62] and DLB can further be classified into two categories namely Distributed and Non-distributed [58].

Load balancing algorithms have the following objectives [58]:

- High Performance - to obtain higher comprehensive development in system performance at a moderate cost, e.g., lowering down the process time while keeping reasonable suspensions.
- Job Equivalence - to deal with the jobs in the system evenly, irrespective of their sources.
- Fault Tolerance - to have performance perseverance under partial breakdown in the system.
- Scalability - to have the capability to handle any modifications or expansion in the distributed system architecture.
- System Stability - to maintain the capability to handle critical conditions like unexpected flow of incoming jobs without performance degradation as well as reducing the waiting time among overall jobs.

An important component of a dynamic load balancing algorithm is its policies, which can be categorized as information policy, transfer policy, selection policy and location policy. The information policy decides when, from where and what load information has to be collected. The transfer policy determines the sending or the receiving resource of the workload. The selection policy selects the appropriate workload to transfer and the results of transfer policy are used by the location policy to discover an appropriate match for a sending or receiving resource [57] [61] [63] [64] [59].

To implement the above mentioned policies global and local strategies are followed. In global schemes, the judgment of load balancing is taken via global information, for example, by simultaneity of processors in delivering their performance profiles to the load balancer whereas the local schemes exhibit similar behaviour as of global equivalents, but profile data is reciprocated locally inside the group only. The global strategies are further divided into Global Centralized DLB & Global Distributed DLB and the two local strategies are Local Centralized DLB and Local Distributed DLB [65].

This section highlighted various cloud enabling technologies like virtualization and load balancing. The next section elucidates the crucial challenges that lie in the way of successful adoption of the cloud technology.

1.4 Cloud Computing Challenges

Although cloud computing has been broadly embraced by the IT sectors, still many existing issues are there, which have not been fully addressed as yet. Some of the key

challenges are [5] [6]:

1.4.1 Virtual Machine Migration

The emergence of cloud computing and the virtualization support offered by it, has further corroborated the efforts for realizing energy efficient computing. It has been observed that the virtualized cloud data centers require lesser energy as compared to the non-virtualized ICT data centers. The extended facility of migrating the running VMs without any perceptible downtime, from the massively-laden nodes to the lowly-laden nodes, helps to manage the workload to minimize energy consumption. The decrease in the consumed energy is due to the improved node utilization that results from a well-adjusted distribution and execution of workload on the nodes. The composed distribution of the workload among the nodes prevents node over-utilization that would have otherwise occurred. The optimally utilized nodes consume low energy in comparison to the over-utilized or under-utilized nodes [66]. The under-utilization of a node indicates that it is sitting idle while the over-utilization of a node means that it is running tasks beyond its capability. The concept of dynamically and transparently migrating VMs from one host to the other, to find the best target host is known as Live migration [67]. Apart from this, the key benefit of VM migration is the identification of hot-spots in the data centers [13]. The over-utilized nodes are the hot-spots and their identification helps to lower energy consumption by migrating their load to the less utilized nodes, leading to green cloud data centers [68].

Presently, perceiving workload hotspots and instigating a migration, lacks the swiftness to counter unexpected workload variations, which is a major challenge. The hotspots can be avoided by VM migration. VM migration enabled by virtual environment, can deliver substantial benefits in cloud paradigm like balancing load and enabling highly responsive provisioning in data centers [16] [45] [4] [13].

1.4.2 Server Consolidation

For the upsurge of resource utilization and to reduce the consumed energy, server consolidation is an efficient process. The server consolidation also termed as task consolidation, is the method to assign n jobs to r cloud resources without the violation of time constraints, thereby maximizing utilization of the resources and minimizing the consumed energy. The energy occupied by the under-utilized resources in a cloud paradigm, brings about a considerable quantity of absolute energy use. A resource

distribution approach, considering resource utilization, prompts to improved energy efficiency. It further augments with the virtual environment which enables virtual machine migration to balance the load across all the nodes in data centers and to avoid hotspots [66] [13].

Server consolidation is a big challenge as it may affect the application performance and resource congestion if a server is maximally consolidated. It is imperative to perceive the timely variations in VM resource usage for effectual consolidation of the servers. Also, there is a need to rapidly respond to the congestions of the resources. An individual server is exclusively used by Live VM migration technology to unite various VMs of less used servers. Thus, the less used servers are deviated to energy-saving state [69] [70].

1.4.3 Energy Management in Cloud

Improvising energy efficiency is additional dominant matter in cloud paradigm. The main concern is to fulfil government regulations and environment criterions, instead of only dropping down the energy cost in the data centers. Energy-aware job scheduling and server consolidation are the main approaches by which the energy consumption of unused machines can be scaled down. Energy-efficient network protocols and infrastructures are also gaining importance. But the most important agenda is to attain energy savings as well as application performance [45] [66] [13].

Energy efficiency is important for the future ICT too. The advancements and pervasiveness of ICT, combined with energy costs hike and the requirement for GHG emissions reduction, also calls for energy-efficient solutions to curtail the comprehensive energy consumption of computation, storage and communications [45]. Cloud paradigm promotes application of data center resources in ICT services [4].

Green computing [71] [72], is the method of application of policies and measures to enhance the efficacy of computing resources that leads to minimization of the consumed energy and the influence of their utilization on the environment. [73] [24] [33] [41].

There is a necessity for the swift and expansible access to high-end computing capacities due to the high recognition of High Performance Computing (HPC) [74]. This kind of environment is provided by cloud computing technology by easing the users to use data and services from cloud anyplace in the world on demand and pay basis [7]. This technology consists of data centers facilitated by speedy computer networks which can execute services effectually on the Internet rather than on local

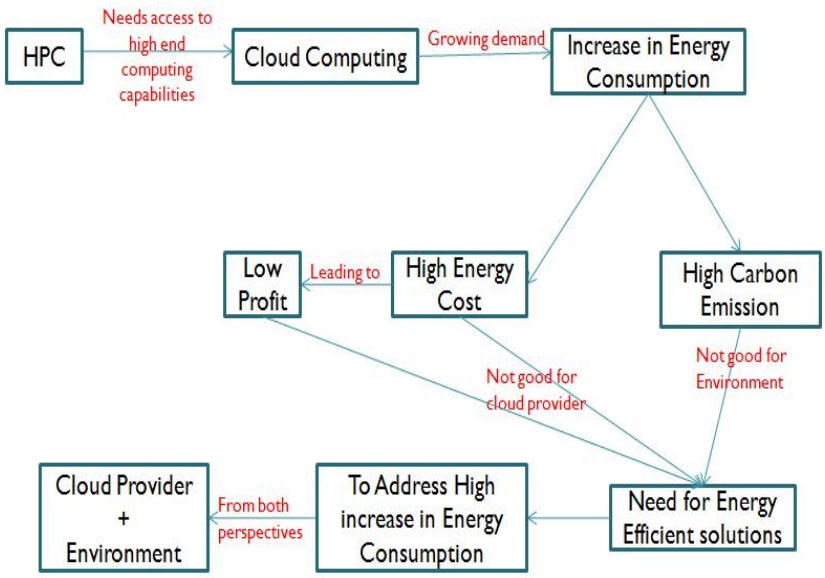


Figure 1.6: Green Computing in Clouds

and/or PCs. Also, hosting and operating these services on the cloud is inexpensive [3]. Owing to exponential evolution of cloud paradigm & its ability to provide high-end computing capabilities, it has widely been embraced by HPC users and ICT industry, resulting in a swift enlargement of data-centers. This enlargement has triggered the energy use as well as its cost, thereby, reducing gain surplus of cloud vendors. This also influences surroundings as carbon footprints [7] [75]. The association of energy consumption and carbon emission has raised a concern of energy-efficiency improvement for both industry and society to accomplish green computing [72] [13]. So, there is a dire need to find energy-aware solutions to curtail energy consumption of the cloud data centers [33] [45] [41].

Energy-efficient solutions are required (as shown in Figure 1.6) which are able to remark the upsurge in the consumed energy in the perception of both the cloud provider and the environment, thereby achieving green computing.

1.4.4 Load Balancing

One of the prime demands in cloud computing is load balancing. Load balancing in cloud paradigm is used to obtain higher level of user satisfaction and resource utilization ratio. Resource utilization can be enhanced with suitable load balancing, causing added minimization in energy consumption and carbon emission rate, that is the need of the

hour in the area of cloud computing [71] [16] [58].

"Load Balancing in clouds is a mechanism which distributes the excess local workload evenly to the whole cloud to achieve a high user satisfaction and resource utilization ratio [76]." It is required for scalability, high availability and high performance in clouds. By dynamically balancing the load among the servers, hot-spots can be avoided and resource utility levels can be improved.

Despite the above mentioned factors, load balancing is an important requisite to attain green computing in clouds. The following three factors aid in achieving the same.

- Curtailing Energy Consumption - The consumed energy is curtailed by preventing overheating. This is done by harmonizing the workload over different nodes in a cloud.
- Minimizing Carbon Emission With the attainment of load balancing in cloud environment, the energy consumption is reduced, which leads to the minimization of the carbon emission rate. This aids in realizing green computing.
- Reducing Cooling Requirements - Cooling requirements of huge data centers are also huge, due to the enormous amount of energy consumption in these data centers. So, if the consumed energy is reduced, the cooling requirements also go down, leading to further reduction in energy consumption, thereby achieving green computing.

Hence there is a prompt requirement to design and develop an energy-efficient load balancing technique in clouds to achieve green computing.

This section summarized the dominant issues in the way of cloud adoption. The following section describes load balancing and energy management in cloud environment, which have been chosen as the area of research for this thesis.

1.5 Research Motivation

Data centers are often provisioned to handle peak loads, which can result in low resource utilization and wastage of energy. Resource utilization directly relates to energy consumption, so it should be optimized in order to save energy [71] [16] [66] [13]. Cloud computing's potentiality for convincing energy savings has so far been aimed at hardware features. Contrarily, software systems can also be improved at development time by postulating the energy characteristics [45].

In cloud paradigm, load balancing is needed for the redistribution of the workload among nodes to prevent a state of some nodes being over-laden while others being lightly-laden or even idle. The workloads should be mapped to various resources under the constraint of energy optimization [45]. With efficient load balancing, resource utilization can be enhanced which can further reduce energy consumption thereby reducing carbon emissions and cooling requirements of cloud data centers. Every operational physical node in a cloud, produces heat. When a specific node is unreasonably used, hot-spots can appear in a given data center. Therefore not only the aspect of load has to be considered, but also the heat generated by a service has to be measured and accounted for to avoid these hot-spots [45].

As a result, there is a necessity for the development of a near optimal load balancing technique that is capable of improving the performance of cloud computing. This technique should be able to balance the workload over the minimum number of required nodes along with the maximum resource utilization. This should lead to the reduction in the energy consumption, thereby dropping the carbon emissions and cooling requirements of the cloud data centers, to an extent which can help to achieve green computing.

Due to the factors outlined above, the energy-aware load balancing for cloud computing environment has been the motivation behind this work. The following section details the contributions of the thesis.

1.6 Thesis Contributions

This research contributes in the following ways:

- It critically analyzes the detailed literature of various existing load balancing techniques along with a detailed study of the work in the area of resource utilization and VM migration techniques, prevalent in the cloud computing environment. It also provides a qualitative comparison of the existing bio-inspired approaches with respect to their parameters like convergence, premature convergence, service, etc.

- To address the challenge of resource under-utilization, an Energy-aware Resource Utilization (ERU) model based on ABC optimization approach, has been proposed for the cloud computing environment. It increases the utilization of server resources, reduces energy consumption and heat dissipation, hence contributes directly to green computing.

- To address the issue of hot-spots and VM migrations, the Firefly Optimization based Energy-aware VM migration (FFO-EVMM) technique has been proposed to achieve energy efficiency in the cloud data centers.
- The credibility of the proposed ERU model and FFO-EVMM approach has been justified by executions in the CloudSim Simulator.
- An Energy-aware Load Balancing (ELB) model has been proposed that makes energy-aware load balancing decisions from both the provider and the user perspectives. Two ELB techniques, namely ELB(RU) & ELB(w/o RU), have also been proposed, to balance the system load, thereby enhancing resource utilization and performance levels, hence reducing energy consumption of the cloud data centers.
- A case study has been conducted at BSNL, Chandigarh, to test and validate the proposed ELB techniques. The algorithms have been empirically evaluated by comparing them with the existing load balancing algorithms. The obtained results illustrate the efficacy of both the approaches.
- Statistical analysis of the results has been performed in order to assess the accuracy of the estimated performance indices by using coefficient of variation. The coefficients of variation have been calculated for resource utilization, energy consumption and performance results achieved by the proposed ELB techniques and existing load balancing algorithms.
- By reducing the consumed energy, all the proposed approaches indirectly reduce carbon emissions and cooling requirements of the cloud data centers, leading to further reduction of energy and hence achieve green computing.

This section provided an insight into various contributions offered by the thesis. The next section presents the details of the organization of this thesis.

1.7 Thesis Organization

After the introduction to the thesis in Chapter 1, the rest of the thesis is structured as follows:

Chapter 2 presents the literature survey in the area of load balancing, resource utilization and VM migration techniques. This chapter describes load balancing and its basics, followed by a detailed survey of existing load balancing techniques. It also lays out a discussion on the existing work of various authors for the efficient utilization

of cloud resources pursued by an up-to-date survey of a wide range of VM migration techniques. Finally, it outlines various bio-inspired techniques and concludes with the objectives of the thesis. Chapter 2 is partially derived from:

- Kansal N.J. and Chana I., *Cloud Load Balancing Techniques: A Step Towards Green Computing*, International Journal of Computer Science Issues (IJCSI), IJCSI Publication, ISSN (Online): 1694-0814, Vol. 9, Issue 1, No 1, pp. 238-246, January 2012, **cited** by 127.

- Kansal N.J. and Chana I., *Existing Load Balancing Techniques In Cloud Computing: A Systematic Review*, Journal of Information Systems and Communication (JISC), Bioinfo Publications, ISSN: 0976-8742, E-ISSN: 0976-8750, Vol. 3, Issue 1, pp. 87-91, March 2012, **cited** by 46.

Chapter 3 proposes an Energy-aware Resource Utilization (ERU) model that is based on the ABC optimization approach. It analyzes the requirements, architecture and the components of the proposed approach along with a discussion on its mode of operation in the model. It moves on to presenting the Artificial Bee Colony (ABC) optimization technique which is the basis of the proposed ERU model and gives the detailed design, algorithm and the flowchart of ERU. This chapter further depicts the performance analysis of the ERU model over the existing FFD [77] & ACO [78] approaches, by using the CloudSim simulator. Chapter 3 has been derived from :

- Kansal N.J. and Chana I., *Artificial bee colony based energy-aware resource utilization technique for cloud computing*, Concurrency and Computation : Practice and Experience (CCPE), Wiley Online Library, Vol. 27, Issue 5, pp. 1207-1225, May 2014, DOI:10.1002/cpe.3295, **cited** by 9.

Chapter 4 proposes a FireFly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) technique that performs the live migration of VMs from one active node to the other active node. Initially, it presents the Firefly optimization technique which is the basis of the proposed VM migration technique, followed by the design of FFO-EVMM technique. Later, it provides the detailed description of the proposed algorithm along with its pseudocode and the flowchart. This chapter, further presents the verification and the implementation of the proposed FFO-EVMM technique over FFD & A in the CloudSim toolkit. Chapter 4 has been derived from :

- Kansal N.J. and Chana I., *Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach*, Journal of Grid Computing (JoGC),

Springer, Vol. 14, Issue 2, pp. 327-345, February 2016, DOI:10.1007/s10723-016-9364-0, **cited** by 2.

Chapter 5 proposes an Energy-aware Load Balancing (ELB) model that makes energy-aware load balancing decisions from both the provider and the user perspectives. This chapter initially presents various modules, working phases and the data flow representation of the ELB model. It then proposes two ELB techniques, namely ELB(RU) & ELB(w/o RU), that effectively balance the load, thereby enhancing the resource utilization and performance levels, hence, curtailing the energy consumption of cloud data centers. Further, it provides the detailed description of the proposed approaches along with their algorithms. Chapter 5 has been partially derived from :

- Kansal N.J. and Chana I., *An empirical evaluation of an energy-aware load balancing technique in cloud data center*, Journal of Cluster Computing (JoCC), Springer.

Chapter 6 demonstrates the verification details, experimental setup and testing results of the proposed ELB techniques. The proposed ELB techniques have been tested and validated by carrying out a case study of BSNL data center, Chandigarh. The performance of the proposed ELB (RU) technique has been compared with the three standard algorithms, FFD [77], ACO [78] & RR whereas the performance of the proposed ELB (w/o RU) technique has been compared with ELB (RU), FFD, ACO & RR. The results of the proposed algorithms with variation in workloads, CPU & memory utilization levels & number of VM migrations have been analyzed which vary in the cloud environment. Also, all the proposed algorithms have been evaluated with respect to both the energy consumption levels and the performance. The testing results demonstrate that the proposed solutions can effectively be used to address the low resource utility and high energy consumption challenges to establish an energy-aware cloud data center. Chapter 6 has been derived from :

- Kansal N.J. and Chana I., *An empirical evaluation of an energy-aware load balancing technique in cloud data center*, Journal of Cluster Computing (JoCC), Springer.

Chapter 7 gives the concluding remarks on the research by highlighting the significant contributions of the work done. Future directions of the implemented research work have also been presented towards the end of this chapter. The chapter concludes the thesis, by explaining briefly, the outcome of each of the chapter.

Chapter 2

Literature Survey

The previous chapter introduced cloud computing paradigm giving an insight into its essential characteristics, various service and deployment models. Besides presenting cloud components, it portrayed the necessity of achieving energy efficiency in cloud data centers by citing the impact of a cloud in minimizing energy demand.

Clouds being an outgrowth of previous distributed paradigms, require novelty in load balancing capabilities along with efficient resource utilization and energy management. Server consolidation improves resource utilization by uniting several VMs onto a single server and VM Migration helps in load balancing and evading hot-spots in data centers thereby dipping energy consumption levels. This chapter conducts an extensive analysis of existing load balancing, resource utilization and VM migration techniques that are being used to accomplish load balancing task. Efficiency of load balancing directly relates to the compliance with user satisfaction level, hence, development of new techniques is essential for performing load balancing in an energy-aware manner to enhance resource utilization, performance levels and energy efficiency of cloud data centers.

This chapter introduces load balancing and its basics, followed by a detailed survey of existing load balancing techniques. Then it lays out a discussion on the existing work for efficiently utilizing the cloud resources pursued by an up-to-date survey of a wide range of VM migration techniques. Finally, it outlines various bio-inspired techniques and the concluding section pens down the objectives of the thesis.

2.1 Load Balancing

The interconnected resources operate separately or in teamwork with each other in case of distributed environment. All these combined resources have some initial workloads assigned to them individually and can have divergent processing ability [79].. This workload is consistently allocated to all the above resources based on their computational speed to minimize the time needed to perform all the workloads [56] and thus the need of the load balancing arises. "Load balancing is a mechanism to distribute the dynamic workload evenly to all the nodes. It is required to achieve a high user satisfaction and resource utilization ratio [76]". It also helps in preventing bottlenecks of the system which may occur due to the load imbalance. Load balancing facilitates the continuation of the service, by implementing fair-over and ensures that every computing resource is distributed efficiently and fairly [16] [80] [81]. In other words, the load balancing distributes the tasks evenly to all the processors so that the condition, where some processors are overloaded while others are lightly loaded or even idle, could be avoided. It ensures the maximal utilization of available resources which in turn improves the response time of applications submitted by the user [57] [58] [59]. The taxonomy of different existing LB schemes as presented in the literature, is shown in Figure 2.1 & the details are discussed in the subsequent sub-sections.

2.1.1 Load Balancing Categories

As per the existing literature, the load balancing algorithms can be classified into the following categories:

Static Load Balancing (SLB): In static load balancing, assignment of tasks to processors is done deterministically or probabilistically at compile time using a prior knowledge about the tasks and the system [60] [61]. SLB can further be classified into [62]:

Optimal SLB - The entire statistics about the system state and the resource requirements is needed for optimum workload distribution.

Sub-Optimal SLB In case of the non-existence of the optimal solution, sub-optimal procedures are applied to escort a load balancing process. These procedures depend on rules-of-thumb and empirical methods.

Dynamic Load Balancing (DLB): In DLB, load distribution decision is made at run-time by making use of the current processes and system state information [60]. DLB can further be classified into two categories [58]:

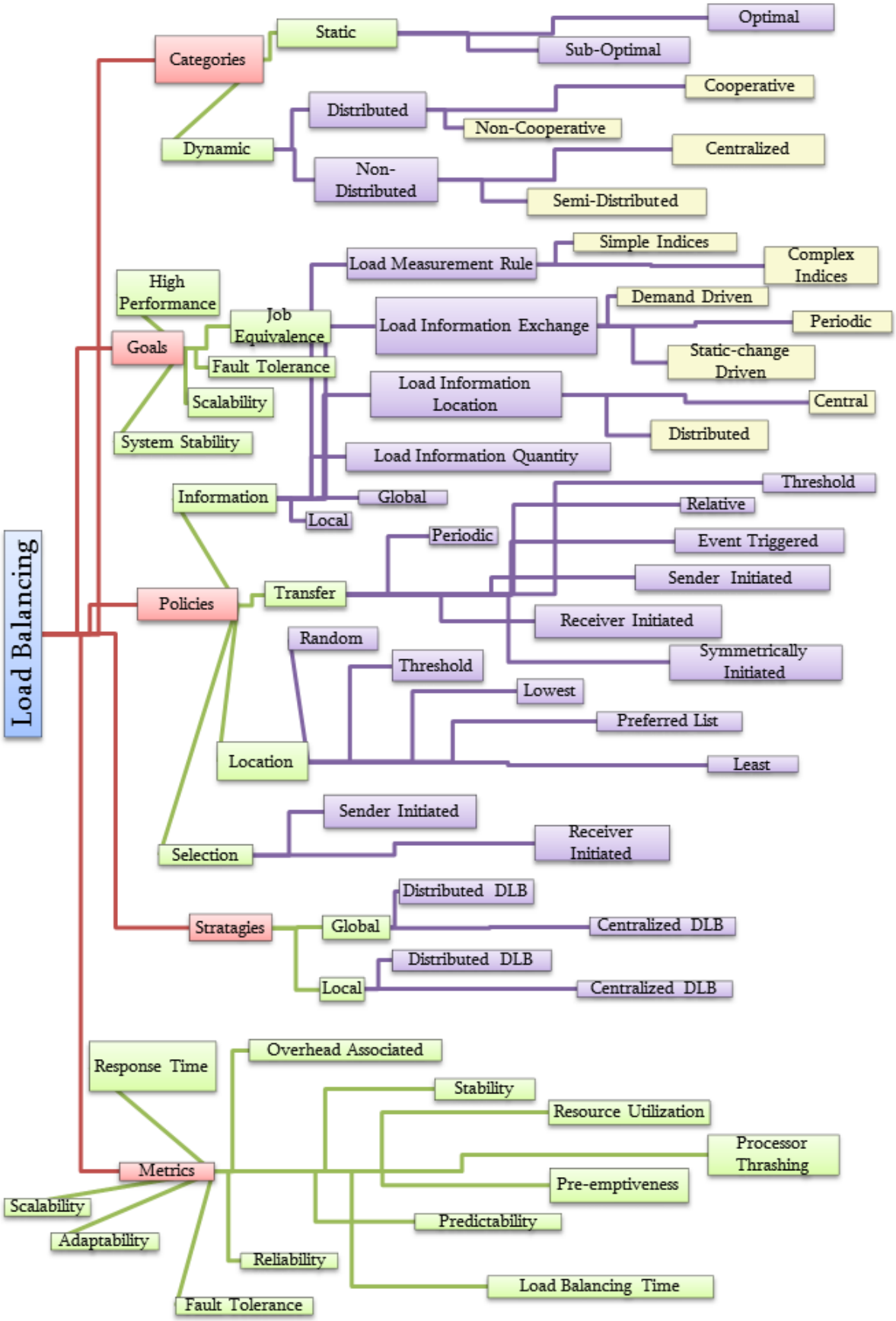


Figure 2.1: Load Balancing Taxonomy

Distributed All system nodes are accountable for load balancing in a distributed system. These nodes interact in two ways- cooperative and non-cooperative. In a cooperative form, nodes work together to realize a global target, e.g., to enhance the systems response time. Whereas, in a non-cooperative form, every node works individually in the direction of a local goal, e.g., to improve a local tasks response time.

Non-distributed The accountability of load balancing in a non-distributed system is either of a single or some nodes but not of all nodes. It can also be achieved in two ways- centralized and semi-distributed. In the centralized method, the load balancing is the responsibility of the central node. The remaining nodes interact with the central node but not with each other. In the semi-distributed method, nodes are segmented into groups. Load balancing within each group is centralized and the responsibility of load balancing for the whole distributed system is distributed among the central nodes of each group.

2.1.2 Load Balancing Goals

Some of the major goals of load balancing algorithms are [58]:

High Performance - to obtain higher comprehensive development in system performance at a moderate cost, e.g., lowering down the process time while keeping reasonable suspensions.

Job Equivalence - to deal with the jobs in the system evenly, irrespective of their sources.

Fault Tolerance - to have performance perseverance under partial breakdown in the system.

Scalability - to have the capability to handle any modifications or expansion in the distributed system architecture.

System Stability - to maintain the capability to handle critical conditions like unexpected flow of incoming jobs without performance degradation as well as reducing the waiting time among overall jobs.

2.1.3 Load Balancing Policies

One important component of a dynamic load balancing algorithm is its load balancing policies [57] [61] [63] [64] [59]. These policies help in collecting and managing system status information and can be classified as shown in Figure 2.1.

Information Policy: Information Policy decides when, from where and what load information has to be collected. It can be further divided into the following types:

Load measurement rule - Load of several nodes in a distributed system is measured by a load index metric. Various metrics can largely be distributed into two main groups - simple and complex.

Simple indices The load taken into account is that on the CPU, including CPU queue length, average CPU queue length over a certain period, extent of available memory, context switch rate, system call rate and the CPU utilization.

Complex load indices - These comprise of a number of metrics, each linking to only one resource, such as CPU, disk, memory and network. The metrics that create the load index can be joined to contribute a single load value. These can be denoted as a tuple comprising of a number of components, one per metric.

Load information exchange policies - These policies can largely be grouped into three categories: demand driven, periodic and state-change driven.

Demand driven - In demand driven policies, the load information is exchanged only when it is required.

Periodic - In periodic policies, the load information is collected periodically.

State-change driven - In state-change driven policies, the dissemination of load information is triggered by a change of the load to a certain degree.

Load information location policies - Either a central storehouse can be adopted to maintain the load statistics of all the nodes of a system or a distributed approach can be used where every node gathers data regarding the load state of the other nodes of the system.

Policies for load information quantity - Either collect the load information of all the processing nodes in the system i.e. globally or of the node in question i.e. locally.

Transfer Policy: Transfer Policy determines the sending or receiving resource of the workload. These can further be classified as:

Threshold transfer policies - may decide that a source/target initiates a transfer of workload to a target/source if the load of the source/target exceeds/falls a certain threshold.

A relative transfer policy - considers the difference between the loads of targets in a domain. If the loads differ more than a certain threshold, workload can be transferred.

Periodic transfer policy - periodically checks if the nodes state qualifies for a workload transfer.

Event-triggered transfer policies - initiates workload transfer either due to state change of any node, or due to arrival of new workload at a source.

Sender-initiated policies - heavily loaded node decides to transfer some workload.

Receiver-initiated policies - a lightly loaded node tries to receive some workload from heavily loaded nodes.

Symmetrically-initiated policies - both senders and receivers may initiate a workload transfer.

Location Policy: Location Policy uses results of the transfer policy to find a suitable partner for a sending or receiving resource. Five typical policies are:

Random policies - A transferal companion is nominated at random, and its load-state is overlooked.

Threshold policies - A node is selected as a potential partner if its load index is either less than or equal to a preset threshold value.

Lowest policies - A node is selected as a partner if its load index is zero.

Preferred list - Each node orders all other nodes into a preferred list and the first node of this list is selected to transfer the load.

Least policies - A node with the smallest load index is selected as a partner.

Selection Policy: Selection Policy selects appropriate workload for transfer. It can be of two types:

Sender-initiated - In sender-initiated policies, busy nodes choose tasks to transfer to another node.

Receiver-initiated - In receiver-initiated policies, lightly loaded nodes inform the potential senders, about the types of tasks they are willing to accept.

2.1.4 Load Balancing Strategies

Following load balancing strategies are used to implement the load balancing policies [65].

Global Strategies: In global methods, the load balancing judgement is made using global information, i.e., all the processors harmonize and direct their performance summaries to load balancer. These are further divided into:-

Global Centralized DLB (GCDLB) - The load balancer, positioned on a centralized processor directs the processors to send the load to other processors, specifying the receiver and the amount of load to be transferred.

Global Distributed DLB (GDDLB) The load balancer, imitated on all the processors broadcasts the summary statistics to all the processors, thereby eradicating the necessity to direct the instructions.

Local Strategies: These are same as their global counterparts, except that summary data is swapped locally inside a group only. The two local strategies are:-

Local Centralized DLB (LCDLB) - There is only one master load balancer, in place of one per group.

Local Distributed DLB (LDDLB) There is a full replication of the load balancer.

2.1.5 Load Balancing Metrics

The various metrics identified from the existing load balancing techniques are [62] [63] [64]:-

Overhead Associated It defines the amount of overhead incurred while realizing a load-balancing approach. It consists of overhead due to movement of tasks, inter-processor and inter-process communication.

Resource Utilization - is practiced to inspect the utilization of resources.

Processor Thrashing It occurs, when the maximum number of the processors are busy in relocating the processes without realizing any valuable work in an endeavour to appropriately schedule the processes for better performance.

Pre-emptiveness - It is associated with the inspection of the executing jobs, whether they can be moved to other nodes or not.

Predictability It is associated with the deterministic or nondeterministic factor to forecast the result of the algorithm.

Adaptability It is used to inspect whether the algorithm can adjust to varying circumstances.

Stability It is associated with interchanging the current workload state statistics amongst processors.

Scalability It is the capability of an algorithm to accomplish load balancing for a system with any finite number of nodes

Response Time It is the extent of time taken to respond by a particular load balancing approach in a distributed system.

Load Balancing Time - is the amount of time that elapses between the job arrival time and the time at which the job is finally accepted by a node.

Reliability - is to determine the reliability of any approach in case of a node failure.

Fault Tolerance It is the capability of an approach to execute uniform load balancing despite of random node failure.

After reviewing the basics of load balancing, the next section portrays a detailed survey of various non-energy-aware and energy-aware load balancing techniques.

2.2 Existing Load Balancing Techniques in Clouds

Load balancing in clouds, distributes the workloads evenly across all the nodes. By doing this, a situation could be prevented where some nodes are overloaded while others are lightly loaded or even idle, thereby avoiding hot-spots, ensuring maximum resource utilization, hence improving performance by reducing response time of cloud users' jobs [57] [58] [59]. These techniques can be classified into two categories namely, non-energy-aware & energy-aware load balancing techniques. As the name suggests, the non-energy-aware techniques do not consider the energy consumption factor, which is in contrast to the energy-aware techniques, that consider the energy consumption as one of the primary goals to achieve green computing.

2.2.1 Non-energy-aware Load Balancing Techniques

The following load balancing techniques are currently prevalent in clouds :

H. Mehta et al. [82] have proposed a new content aware load balancing regulation named as "Workload and Client Aware Policy (WCAP)". It uses a "Unique and Special Property (USP)" to specify the USP of the requests as well as computing nodes. The USP aids scheduler for the decision making of the appropriate node for processing the requests. This approach is executed in a distributed way with low overhead. It increases the searching optimization of the system thereby minimizing the idle time of the computing nodes and refining their utilization.

A. M. Nakai et al. [83] have designed a new server-based load balancing regulation for web servers which are decentralized globally. It uses a middleware to implement a protocol that redirects the requests to the closest remote servers without overloading them. This assists in decreasing the service response times. A heuristic is also used to aid overload sustenance of the web servers.

Y. Lua et al. [84] have given a "Join-Idle-Queue" load balancing algorithm for dynamically expandable web services. This algorithm provides enormous load balancing with distributed dispatchers by. The load balancing work is detached from the critical path of request processing, thereby effectively reducing the system load. Further, no communication burden is obtained at job arrivals and actual response time is also not increased.

X. Liu et al. [85] have offered a "lock-free multiprocessing load balancing solution" that prevents the use of shared memory as compared to other multiprocessing load balancing criterions. This has been achieved by reforming the Linux kernel. The overall performance of load balancer is improved in a multi-core environment by executing various load-balancing channels in one load balancer.

J. Hu et al. [86] have suggested a genetic algorithm based scheduling strategy on load balancing of VM resources. This strategy uses the historical and current state of the system, to compute the influence of required VM resource allocation on the system. By selecting the least-affective solution, it supports in analysing the problem of load-imbalance and expenses of migration, thereby attaining reasonable resource utilization.

A. Bhadani et al. [87] have presented a "Central Load Balancing Policy for Virtual Machines (CLBVM)" that proportions the load equally in cloud computing. The

system performance is enhanced but fault-tolerant systems are not considered.

H. Liu et al. [88] have advised a "Load Balancing Virtual Storage Strategy (LBVS)" that offers a high extent "Storage as a Service model" based on cloud Storage. A three-layered architecture is used to accomplish storage virtualization. Two components are used for load balancing. The efficacy of concurrent access is boosted by the use of replica balancing, thereby decreasing response time and increasing disaster recovery capacity. Additionally, supporting adaptability and power of the system.

Y. Fang et al. [89] have talked over a two-level task scheduling technique based on load balancing to meet active needs of users and get a high resource utilization. Here, load balancing is acquired by first assigning jobs to VMs and then VMs to resources, thereby increasing job response time, resource utilization and overall performance of the cloud environment.

M. Randles et al. [90] have examined a "decentralized honeybee-based load balancing technique" acquiring global load balancing through local server activities. System performance has been improved with amplified system multiplicity making it suitable for varied service types. The throughput remains the same with the growth of system size [91] [92].

They have also presented a decentralized and expandable load balancing technique. This technique practices random sampling of the system domain to realise self-organization, thereby obtaining load balancing over system nodes. The system performance has been enhanced with comparable resources, consequentially improving throughput by proficiently exploiting the amplified resources. Whereas the performance of the system lowers with variation in resources [91] [92] [93].

Further, the authors have examined a "self-aggregation load balancing technique" for the optimization of job allocations. This is obtained by joining comparable services using local re-wiring. In this, the enhancement in the number of resources, results in the improved system performance. Moreover, the effective usage of these resources helps in increasing the throughput. Again, the upsurge of system diversity degrades the system performance [91] [92] [94].

Z. Zhang et al. [76] have projected a "Load Balancing mechanism based on Ant Colony and Complex Network Theory (ACCLB)" in a cloud federation. Superior load balancing is reached by using small-world & progression-free features of a composite network. The system performance is improved by overpowering heterogeneity, fault tolerance and achieving decent scalability.

S.-C. Wang et al. [95] have offered a two-phase scheduling approach by integrating "Opportunistic Load Balancing (OLB) and Load Balance Min-Min (LBMM)" scheduling algorithms. OLB aids in fruitful realization of load balancing whereas LBMM is exploited to diminish the overall completion time by lessening the task execution time. This collective methodology benefits in efficient resource utilization and enhanced work efficiency.

An event-driven load balancing technique has been advised by V. Nae et al. [96] for real-time "Massively Multiplayer Online Games (MMOG)". The load balancing activities of the game period have been produced, after evaluating the constituents of capacity events in terms of resources and game sessions global form. A game session is being scaled up/down on several resources in accordance with the flexible user load.

R. Stanojevic et al. [97] have offered "CARTON", which is a process to regulate cloud. This process combines Load Balancing (LB) and Distributed Rate Limiting (DRL). The tasks are evenly dispersed on various servers by using LB to diminish the associated costs. However, a reasonable resource allocation is preserved using DRL. To maintain the equal performance levels of servers, DRL is acclimatized to the varied capacities of the servers. As there is little computation and communication burden, therefore this approach is implemented with ease and simplicity.

Y. Zhao et al. [98] have addressed the problem of intra-cloud load balancing by adaptive live migration of VMs. VMs are assured to be migrated from high-cost hosts to low-cost hosts by using a load balancing approach known as "COMPARE_AND_BALANCE". The load balancing is achieved and VMs' migration time is reduced by considering a load balancing model.

A. Singh et al. [99] have designed a load balancing technique called "VectorDot" that deals with the hierarchical intricacy of the cloud data-center and multi-dimensionality of resource loads, over servers, storage and network switches. The nodes are discriminated based on their requirements by using the dot product. This helps in eliminating excess loads from servers, switches and storage nodes.

Table 2.1 shows the summary of existing non-energy-aware load balancing techniques in cloud computing. This review identifies the techniques, their environment, description and characteristics in accordance with the details provided in each of the selected papers.

Table 2.1: Analysis Table of Existing Non-energy-aware Load Balancing Techniques

Technique	**Environment**	**Description**	**Characteristics**
Decentralized content aware [82]	Distributed computing	-Uses a unique and special property (USP) to map requests & computing nodes -Uses content information to narrow down the search	-Improves the searching performance hence increases overall performance -Reduces idle time of the nodes
Server-based LB for Internet distributed services [83]	Distributed web servers	-Uses a protocol to limit redirection rates to avoid remote servers overloading -Uses a middleware to support this protocol -Uses a heuristic to tolerate abrupt load changes	-Reduces service response times by redirecting requests to closest servers without overloading them
Join-Idle-Queue [84]	Cloud data centers	-First assigns idle processors to dispatchers for availability of idle processors at each dispatcher -Then assigns jobs to processors to reduce average queue length of jobs at each processor	-Effectively reduces the system load -Does not increase actual response times -Incurs no communication overhead at job arrivals
Lock-free multi-processing solution for LB [85]	Multi-core	-Runs multiple load-balancing processes in one load balancer	-Improves overall performance of load balancer
Scheduling strategy on LB of VM resources [86]	Cloud Computing	-Uses Genetic algorithm, historical data and current state of system to achieve best load balancing and to reduce dynamic migration	-Solves the problems of load imbalance and high migration cost
CLBVM [87]	Cloud Computing	-Uses global state information to make	-Balances the load evenly to improve overall

		load balancing decisions	performance -Does not consider fault tolerance
LBVS [88]	Cloud Storage	-Uses Fair-Share Replication strategy to achieve Replica load balancing module which in turn controls the access load balancing -Uses writing balancing algorithm to control data writing load balancing	-Enhances flexibility and robustness -Provides large scale net data storage and storage as a service
Task Scheduling based on LB [89]	Cloud Computing	-First maps tasks to VMs and then VMs to host resources	-Improves task response time & resource utilization
Honeybee Foraging Behavior [90]	Large scale Cloud Systems	-Achieves global load balancing through local server actions	-Performs well as system diversity increases -Does not increase throughput as system size increases
Biased Random Sampling [93]	Large-scale Cloud systems	-Achieves load balancing across all system nodes using random sampling of system domain	-Performs better with high and similar population of resources -Degrades as population diversity increases
Active Clustering [94]	Large-scale Cloud systems	-Optimizes job assignment by connecting similar services by local re-wiring	-Performs better with high resources -Degrades as system diversity increases -Utilizes the increased system resources to increase throughput

ACCLB [76]	Open Cloud Computing Federation	-Uses small-world and scale-free characteristics of complex network to achieve better load balancing	-Overcomes heterogeneity & Adaptive to dynamic environments -Excellent in fault tolerance & Good scalability
(OLB + LBMM) [95]	Three-level Cloud Computing Network	-Uses OLB (Opportunistic Load Balancing) to keep each node busy and uses LBMM (Load Balance Min-Min) to minimize execution time of each task	-Efficient utilization of resources -Enhances work efficiency
Event-driven [96]	Massively Multiplayer Online Games	-Uses complete capacity event as input, analyzes its components and generates the game session load balancing actions	-Capable of scaling up & down a game session on multiple resources according to the variable user load -Occasional QoS breaches
Carton [97]	Unifying framework for cloud control	-Uses Load Balancing to minimize associated cost and uses Distributed Rate Limiting for fair allocation of resource	-Simple & Easy to implement -Very low computation & communication overhead
COMPARE_ AND_ BALANCE [98]	Intra-Cloud	-Based on sampling -Uses adaptive live migration of VMs	-Balances load amongst servers & Reaches equilibrium fast -Assures VM migration from high-cost hosts to low-cost hosts -Assumption of having enough memory with each physical host
VectorDot [99]	Datacenters with	-Uses dot product	-Handles hierarchical &

	integrated server and storage virtualization	to distinguish node based on item requirement	multidimensional resource constraints -Removes overloads on server, switch and storage

From Table 2.1, it can be analyzed that the most of the existing load balancing techniques have mainly focused on reducing service response time and improving performance. A few of these techniques have targeted towards reducing the associated overhead and migration cost and handle fault tolerance. Only a few have considered resource utilization factor. But, none of the techniques has worked towards achieving the high resource utilization, performance and reducing energy consumption levels collectively, thereby enhancing energy efficiency in cloud data centers.

The next subsection portrays a detailed survey of the various energy-aware load balancing techniques prevalent in cloud computing.

2.2.2 Energy-aware Load Balancing Techniques

To reduce the energy consumption in the cloud data centers, a lot of research is being conducted as surveyed in [33] [41]. Many energy-aware techniques have been proposed and devised to overcome the wastage of energy in cloud data centers. Load balancing being an influential energy management technique, is being vigorously investigated to manage the data center energy. This section briefly discusses the existing energy-aware load balancing techniques in cloud computing. The non energy-aware load balancing techniques in cloud computing have been listed and discussed in the above subsection.

Megharaj et al. [100] have recommended a two level centralized scheduling model of load balancing in cloud to overcome the high communication cost of the distributed algorithms and a single point of failure problem of centralized algorithms. This model is hierarchical with the Global Centralized Scheduler (GCS) at the higher level and the Local Centralized Scheduler (LCS) at the next level. The authors have not proposed any energy-efficient solution. They have just said that suitable energy efficient load balancing algorithms for cloud may be designed to demonstrate load balancing to achieve green computing.

Ruzan et al. [101] have formulated a hybrid algorithm using genetic optimization algorithm and Hadoop MapReduce framework to promote the energy efficiency in the

cloud computing platform. The authors have focused on obtaining the minimum optimization value thereby improving a bit of the energy efficiency of servers. Their work considers only CPU utilization and the energy saving parameters and values are implicit.

Anandharajan et al. [102] have focused on finding the best efficient cloud resource by using a co-operative power-aware scheduled load balancing technique for an efficient computational cloud as a solution to the cloud load balancing challenge. The authors have worked towards energy efficiency by considering a hardware approach.

Galloway et al. [103] have projected a power aware load balancing (PALB) algorithm for cloud Computing and Adhikari et al. [104] have devised a double threshold energy aware load balancing algorithm (DT-PALB) which is an extension of the PALB technique. Both the approaches decide the number of active compute nodes based on their utilization percentages thereby providing an adequate availability to compute node resources and decreasing the overall power consumed by the local cloud. The DT-PALB performs better than PALB but both the approaches instantiate the VMs on the least utilized nodes and work towards reducing the number of active nodes to save energy. They have not dealt with enhancing the resource utilization and performance levels, reducing the number of VM migrations to save energy.

Ghafari et al. [105] have suggested a power-aware load balancing method which is a hybrid of artificial bee colony algorithm and the minimal migration time algorithm, to decline the power consumption in the cloud computing, thereby declining the CO_2 production and the operational cost. However, there is a trade-off between the energy consumption and providing high quality of service to the customers. Also, their approach does not consider the energy optimization at the memory level which is a primary source of energy consumption in data centers.

A green solution for cloud computing with load balancing and power consumption management has been proposed by Dalapati et al. in [106]. The authors have emphasised only on the scheduling of jobs for proper load balancing in virtual machines and then putting the idle or unused machines to sleep mode for better power management leading to a green cloud computing solution. They have aimed to implement it practically with a neural network predictor. The energy optimization at the memory level has not been considered.

In [107] [108], Sallami et al. have presented a load balancing solution with neural network to achieve green computing. The artificial neural network predicts the demand

and thus allocates the resources according to the demand thereby achieving the low energy consumption. Their approach does not consider the CPU and the memory as resources explicitly.

The work of RamKumar et al. [109] has offered an adaptive earliest deadline first (AEDF) algorithm to consume cloud virtual resources efficiently and to balance the load effectively. The authors have focused on minimizing the amount of active servers by setting a workload threshold thereby achieving energy savings. Their work does not consider the CPU & the memory utilization separately. The energy saving parameters and values are implicit.

Table 2.2 below presents an analysis of the energy-aware load balancing techniques presently available in cloud computing environment.

Table 2.2: Analysis Table of Existing Energy-aware Load Balancing Techniques

Technique	**Characteristics**	**Drawbacks**
Ruzan et al. [101]	-Hybrid algorithm -Uses genetic optimization algorithm and Hadoop MapReduce framework	-Obtains minimum optimization value -Improves very little energy efficiency of servers -Considers only CPU utilization -Energy saving parameters & values are implicit
Anandharajan et al. [102]	Co-operative power-aware scheduled load balancing technique for cloud	Hardware approach
Galloway et al. [103]	-Power aware load balancing(PALB) algorithm -Decides the number of active compute nodes based on their utilization % & provides an adequate availability of resources	Only reduces active nodes to save energy
Adhikari et al. [104]	-Double threshold energy aware load balancing algorithm (DT-PALB) -Extension of PALB - Performs better than PALB	Only reduces active nodes to save energy

Ghafari et al. [105]	-Power-aware load balancing method -Hybrid of ABC and minimal migration time algorithms -Declines CO_2 production & operational cost	-Trade-off between energy consumption & high quality of service -Does not consider energy consumed by memory
Megharaj et al. [100]	-Two level centralized scheduling model -Overcomes the high communication cost of distributed algorithms -Overcomes a single point of failure of centralized algorithms	-Did not propose any energy-efficient solution
Dalapati et al. [106]	-Green solution with load balancing and power consumption management	- No energy consumption at memory level
Sallami et al. [107] [108]	-Load balancing solution using neural network to predict the demand -Allocates resources according to demand to achieve energy efficiency	Does not consider the CPU and memory as resources explicitly
RamKumar et al. [109]	-Adaptive earliest deadline first(AEDF) algorithm to efficiently use cloud virtual resources and to balance the load -Minimizes active servers to achieve energy savings	-Does not consider CPU & memory utilization separately -Energy saving parameters & values are implicit

After a comprehensive investigation conducted to study various existing load balancing techniques in cloud computing, it can be inferred that most of the techniques have emphasized on the CPU utilization as the sole unit to accomplish energy efficiency. None of the techniques offer energy optimization at the memory level. Only, the number of active nodes have been reduced to save energy and VM migration is not targeted. Further, the values of energy-saving parameters have not been calculated, but are implicit only. Also, some of the techniques have not been analyzed practically. As optimum resource utilization contributes directly towards the energy efficiency, therefore, the next section studies various resource utilization techniques currently prevailing in cloud computing.

2.3 Existing Resource Utilization Techniques

Resource under-utilization is a major impediment in achieving energy efficiency in cloud data centers. It is basically a large-scale optimization problem due to the heterogenous and dynamic nature of cloud resources. Resource utilization allows these under-utilized resources to be used efficiently to reduce their wastage and to curtail energy consumption [110]. To conserve energy, the jobs need to be allocated to the cloud resources in such a way so as to use them efficiently. This section surveys the research work accomplished in this area.

The emergence of cloud computing as a new paradigm, where computing is treated as a utility, has given rise to large data centers [1]. Hardware capacity of these data centers is typically over-provisioned because it is hard to adjust dynamically, resulting in too much hardware, that is highly inefficient at delivering IT services. Most data centers host applications on dedicated servers due to the demands of different users to run their applications in isolation [111] [112]. Thus, these servers are fully utilized only for a fraction of time, while remain under-utilized for other times. This leads to a problem of server sprawl [113], that is multiple servers mostly remaining under-utilized. Under-utilized servers consume more power and require more resources for cooling than required by the average workloads running on them [114] [112] [115]. Hence, these resources that run at average utilization rates of only 5-15% [116] are at the root of energy consumption problem which is a critical issue for IT organizations today. High energy consumption gives rise to another problem of high heat dissipation or CO2 emission, being one of the causes of global warming. Hence, there is a need to develop a solution to make use of these under-utilized resources to save energy [45] [117]. Further, this solution has to be efficient to maximize the resource utilization thereby minimizing energy consumption, hence reducing CO2 emissions and cooling requirements of cloud data centers.

The technologies like server consolidation and virtualization have been used to improve resource utilization and energy-efficiency of clouds. The efficient utilization of resources can be achieved via Server Consolidation (SC) [118], that is a process of aggregating several applications running on multiple servers into a reduced number of physical hosts to increase server utilization. The idle servers can be turned off or put to sleep mode to reduce energy consumption in data centers [13] as components like discs, memory, network devices, also consume energy and a server that seems to remain idle may still use up to 60% of its peak power [45]. SC is enabled through virtualization that is a technology to enable partitioning of one physical node into sev-

eral equivalent (similar cloud distribution) Virtual Machines (VMs) capable of hosting independent operating system instances [111] [119] [120]. This reduces the amount of hardware in use, thereby reducing the complexity of managing the infrastructure. Virtualization further enables one to exploit the power of unused computing resources by consolidating different workloads on a server and hence improves the utilization of its resources [121] [4] [112].

Efficient resource utilization techniques are essential in HPC systems to allocate incoming workloads to efficient system resources while satisfying the performance and energy efficiency goals. Scheduling of computational resources helps to optimize the execution of workloads thereby achieving higher resource utilization levels [IBM About 2013], consequently lowering energy demand. In other words, the level of resource utilization and the amount of consumed energy are closely related as the resources, if under-utilized or over-utilized, demand more energy. Thus, the optimal resource utilization debases the energy consumption levels [66] [122]. Several energy-efficient strategies for resource utilization have been presented as studied in the existing work. These are as follows :

Beloglazov et al. [123] [4] have proposed efficient heuristics for dynamic adaption of VM allocation at run-time according to the current utilization of resources applying live migration to minimize energy consumption. The algorithms do not depend on a particular type of workload. Sharifi et al. [124] have proposed a scheduling algorithm to map VMs onto PMs to maximize the energy reduction while keeping the performance isolation between applications. They have focused on managing energy consumptions and efficient utilization of processor and disk resources using live migration of VMs, which is a costly operation and incurs overhead that directly impacts the response time and energy consumption.

Bobroff et al. [125] proposed and evaluated a dynamic server consolidation algorithm that uses historical data to forecast future resource demand and relies on periodic utilization patterns to reduce the number of physical nodes.

Song et al. [126] have proposed resource allocation to applications according to their priorities in multi-application virtualized cluster. They have also introduced three levels of scheduling to improve the resource utilization resulting in reduced energy consumption. Cardosa et al. [127] have proposed an approach to consolidate VMs according to their pre-defined maximum and minimum resource requirements. Though the allocation of VMs in heterogeneous environments, is power-efficient, yet it requires the knowledge of application priorities and considers only CPU as a resource.

Srikantaiah et al. [69] have proposed an algorithm based on the measured optimal points to consolidate workloads in order to balance energy consumption and performance. They studied the impact of consolidation of multiple workloads with different resource usage on performance, energy usage, and resource utilization. However, their approach is application and workload type dependent.

Kusic et al. [128] have used lookahead control (LLC) method to define the problem of performance and power efficient resource-management in virtualized heterogeneous environments. It performs necessary reallocations depending on the prediction of future requests. Their model requires simulation-based learning and is complex in nature resulting in high execution time.

Authors in [4] [66] [129] [114] have merely focused on reducing the number of active servers by consolidating many tasks onto fewer servers. This may increase the level of complexity and conflicts within consolidated applications thereby increasing energy consumption in the long-term. Lee and Zomaya [66] have presented two energy-conscious task consolidation heuristics, which aim to maximize resource utilization by assigning each task to the resource on which the energy consumption for executing the task is explicitly or implicitly minimized without the performance degradation of that task.

Feller et al. [130] have presented a nature-inspired Ant Colony Optimization (ACO) based workload placement algorithm to compute the placement dynamically according to the current load. This ACO-based approach has been compared to FFD to show energy gains through better server utilization and a reduced number of machines. However, the computation time required to derive the placement and thus the energy spent in computation are significantly higher in ACO-based approach as compared to FFD approach.

Table 2.3 demonstrates an assessment of the existing resource utilization techniques presently applicable in cloud computing environment.

Table 2.3: Analysis Table of Existing Resource Utilization Techniques

Technique	Characteristics	Drawbacks
Beloglazov et al. [123]	-Efficient heuristics for run-time VM allocation -Considers current utilization of resources -Minimizes energy consumption	-Do not depend on a particular type of workload -Does not include power consumed

[4]		by memory & network resources
Sharifi et al. [124]	-Scheduling algorithm to map VMs onto PMs -Maximizes energy reduction -Performance isolation b/w applications -Resources are processor & disk -Live migration of VMs	-Do not consider Memory -Live migration is costly and incurs overhead -Impacts the response time and energy consumption
Bobroff et al. [125]	Dynamic server consolidation	-Forecasted resource demand & relies on periodic utilization patterns -Reduces physical nodes only
Song et al. [126]	-Resource allocation to applications -Three levels of scheduling -Improves resource utilization -Reduces energy consumption	-Requires knowledge of application priorities -Considers only CPU as a resource
Cardosa et al. [127]	-Consolidates VMs -Requires pre-defined maximum and minimum resource requirements	-Application priority knowledge is required -Considers only CPU as a resource
Srikantaiah et al. [69]	-Consolidates workloads based on the measured optimal points -Balances energy consumption & performance	-Application and workload type dependent
Kusic et al. [128]	-Uses lookahead control (LLC) method -Performs reallocations depending on Prediction of future requests -Power and performance management	-Prediction of future requests -Requires simulation-based learning & complex in nature -High execution time
Buyya et al. [4] [66] [129] [114]	-Reduces active servers & consolidates many tasks onto fewer servers	-Increases complexity & conflicts within consolidated applications -Increases energy consumption in long-term
Lee & Zomaya [66]	-Two energy-conscious task consolidation heuristics -Maximizes resource utilization -Energy consumption is explicitly or implicitly minimized	-Alleviates overhead running costs -Increases energy consumption in long-term -Offers no high performance guarantee

	-Performance is not degraded	
Feller et al. [130]	-ACO based workload placement -Dynamic placement according to current load -Energy gains through better server utilization & reduced number of machines	-Computation time is higher -Energy spent in computation is higher -Extreme dependency on workload & resource details, can not handle the nature of tasks

As analyzed by Table 2.3, the discussed resource utilization techniques consider only CPU as a resource. Memory as a resource has not been considered in any of the techniques. Some of the techniques do not guarantee high performance due to their increased complexity and inability to resolve the conflicts within consolidated applications, thereby leading to energy wastage.

After identifying the existing resource utilization algorithms in the area of cloud computing, the next section highlights distinct virtual machine migration techniques that are presently existent in the cloud computing environment. The virtual machine migration technique is required to migrate virtual machines from an over-loaded active node to an under-loaded active node in order to balance the load and to save energy.

2.4 Existing VM Migration Techniques

Cloud computing [131] characterizes a vital step in computing by offering shared computational power of the resources on demand [121]. Being grounded on the fundamental concept of virtualization [132], it has significantly transformed the manner of delivering the IT services with minimized infrastructural requirements. The virtual environment involves the creation of multiple VMs (or virtual servers) on a single physical node. In actual context, the multiple operating systems (OSs) can run on a single OS underlying the same hardware platform. The running of virtual servers minimizes the resource idle time, thus preventing resource under-utilization [38] [133]. Additionally, the reduction in the amount of required hardware, lowers the power needed for operation which consequently cuts down the energy demand. The diminution in the energy demand by ICT sector is highly appreciated in the current scenario of rising energy crisis. Energy efficiency [134] has thus gained prominence in ICT data centers that host massive servers resulting in the induced upsurge of energy consumption levels [135].

The emergence of cloud computing and the virtualization support offered by it, has further corroborated the efforts for realizing energy efficient computing. It has been observed that the virtualized cloud data centers require lesser energy as compared to the non-virtualized ICT data centers. The extended facility of migrating the running VMs without any perceptible downtime, from the heavily-loaded nodes to the lowly-loaded nodes, helps to manage the workload to minimize the energy consumption. The decrease in the consumed energy is due to the improved node utilization that results from a well-adjusted distribution and execution of workload on the nodes. The composed distribution of workloads among the nodes prevents node over-utilization that would have otherwise occurred. The optimally utilized nodes consume less energy as compared to the nodes that are over-utilized or under-utilized [66]. The under-utilization of a node indicates that the node is sitting idle while the over-utilization of a node means it is running tasks beyond its capability. The concept of dynamically and transparently migrating VMs from one host to the other, to find the best target host is known as Live migration [67]. Apart from this, the key benefit of VM migration is identification of hot-spots in the data centers [13]. The over-utilized nodes are the hot-spots and their identification helps to lower the energy consumption by migrating their load to the less utilized nodes, leading to green cloud data centers.

A lot of research is being conducted in the area of cloud computing to reduce the power consumption in data centers as surveyed in [33] [41]. Many different techniques to overcome the power wastage have been proposed and devised with and without VM migration. Live VM migration is being vigorously investigated since long and numerous techniques have been developed to migrate a running VM from one active node to the other active node. It has been observed to be an influential technique for efficiently managing the data center energy. Most of the prevailing VM migration methodologies for energy management in cloud data centers are not straight forward, as primarily they involve VM consolidation or VM placement approaches at the higher level of implementation. This section briefly discusses such techniques.

Feller et al. [136] have put forward a scalable and autonomic VMs management framework that uses a centralized ACO-based VM consolidation algorithm to locally consolidate the VMs. Tarighi et al. [137] have offered a fuzzy decision making based VM migration scheduling algorithm, that discovers the maximally loaded servers and takes a migration decision by using TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) approach. Wood et al. [138] [118] have suggested two gray-box and black-box approaches for virtualized cluster to diminish the hotspots by monitoring

and detecting hotspots and then allowing the live migration of VMs. Marzolla et al. [139] have projected a VM consolidation protocol based on a coarse-grained gossip that apply local VM consolidation by migrating the VMs from the smallest laden node to the greatest laden node. All the above mentioned techniques have considered VM migration in one or the other way, but there is no reflection of energy savings.

Nathuji et al. [119] have designed an architecture for the management of the energy in the virtualized data centers by using VM live migration to consolidate multiple VMs on a single server. Tolia et al. [140] have practiced a short-term VM migration for consolidating workloads and to put the under-utilized servers in the sleep mode. Lim et al. [141] have presented a way of consolidating VMs onto a lesser number of hosts by dynamically migrating virtual machines to save energy in a virtualized environment. A multi-objective profit-oriented algorithm to place VMs has been proposed by Goiri et al. [142]. Performance in terms of SLA violations, energy efficiency and overheads of virtualization have been considered in this algorithm. Ghribi et al. [135] have offered a combination of an exact VM allocation algorithm and an exact VM migration algorithm for reducing the number of nodes and hence to save energy in cloud data centers. Verma et al. [143] [144] have proposed a framework that examines the VM placement algorithms by considering the energy and the migration costs as well as the performance benefit in a virtualized sever cluster, to maximize performance and to minimize energy consumption. Mehta and Neogi [145] have preseneted a ReCon tool to dynamically consolidate servers in data centers. The VMs are mapped to the servers by considering the static and the dynamic costs of physical servers, the cost of VM migration, and the resource consumption data from the history. The work cited here, does not consider energy consumption done by memory which is a prime contender for energy consumption in cloud data centers [33] [41].

An approach for VM consolidation has been offered by Cardosa et al. [127] that agrees to the highest and lowest resource requirements of the VMs to achieve energy-efficiency. Resource utilization is improved and energy consumption is reduced by consolidating several VMs onto a single server. Effectual energy-aware heuristics to allocate VMs dynamically have been advocated by Beloglazov et al. [133] [123] [135]. These heuristics practice live migration of VMs to minimize energy consumption by reducing the number of used nodes and without having required the knowledge of VMs applications [146] [147]. A solution for VM placement and consolidation that is grounded on Bernoulli trials has been proposed by Mastroianni et al. [148] by considering energy and migration cost. Dong et al. [149] have recommended a scheme

to place VMs that meets several resource restrictions to reduce energy consumption by enhancing resource utilization and by saving number of used servers and network elements. All these papers have emphasized on the VM placement techniques and their focus is mainly on reducing the used servers in order to lower the energy levels without considering the reduction in number of VM migrations.

Vu et al. [150] have projected a VM placement algorithm that enhances the performance of communication by decreasing the overall cost of the virtual machine traffic and saves energy by increasing the utilization of the CPUs. Sekhar et al. [151] have designed an energy efficient VM live migration technique based on greedy heuristics to curtail the consumed energy in cloud data centers. Jung et al. [152] have established a framework to optimize the energy consumption by using the live VM migration for consolidating virtual servers and by switching-off the idle servers in cloud data centers. Bila et al. [153] have offered a technique that partially migrates the VMs that are idle and are running on the desktops of the users to a consolidation server to reduce overall consumed energy. Graubner et al. [154] have extended the Eucalyptus cloud management framework to incorporate the support for live migration and consolidation. Xiaoli et al. [155] have presented an energy-aware VM placement algorithm for making cloud data centers more energy-efficient by increasing resource utilization. The resource utilization as well as the energy cost in migrations have been considered in their approach. None of the techniques listed in this paragraph, deals with CPU and memory utilization for lowering the energy consumption. Also, they attempt to diminish the consumed energy without considering the hosts and the VM migrations.

The technique mentioned in [156] mainly targets the reduction in the number of active nodes and the number of VM migrations to cut down the consumed energy in the overall data center. The energy saving parameters and values are implicit i.e no specific values have been given. This techniques does not individually compute the energy consumption of VMs and nodes which is important to keep a track of the energy consumption of each and every VM and node in the cloud data center further helping in analyzing the workload handling capacity of the node.

The authors in [123] advocate different energy-aware heuristics for dynamically allocating VMs in accordance with the current resource utilization. The live migration of VMs is practiced to set aside the free resources that are then switched to the sleep mode, hence cutting down the energy consumption done by them when in idle mode. The main focus for preserving the free resources is to lower the SLA violations and to improve the energy-efficiency of the data center. These heuristics run on varied under-

lying infrastructure and assorted VMs while maintaining the SLA constraints imposed by the users. They have primarily attempted to optimize the energy consumption done by the processor while missing out the energy consumption done by the memory. The memory is one of the most vital elements of emphasis in the power and energy usage optimization in the current scenario [157]. In order to achieve an optimal VM migration and placement, it is important to consider the current utilization of processor and memory which have been observed to be the major power consuming units in a system [33] [41] [157].

Table 2.4 exhibits an evaluation of various existing virtual machine migration techniques in cloud computing environment.

Table 2.4: Analysis Table of Existing Virtual Machine Migration Techniques

Technique	Characteristics	Drawbacks
Graubner et al. [154]	-Extension of Eucalyptus cloud management framework for live migration & consolidation	-CPU & memory utilization not considered -Hosts are not saved -VM migrations are not reduced
Lim et al. [141]	-Consolidates VMs on lesser number of hosts by migrating VMs dynamically -Saves energy	Does not consider energy consumed by memory
Feller et al. [136]	-Scalable and autonomic VMs management framework & uses centralized ACO-based approach to consolidate VMs	No reflection of energy saving
Dong et al. [149]	-A scheme to place VMs meeting several resource restrictions -Reduces energy consumption -Enhances resource utilization	-Focuses on VM placement -Reduces number of used servers only -No details of reducing VM migrations
Nathuji et al. [119]	-Designed an architecture for energy management in virtualized data centers -Consolidates multiple VMs on a single server using VM live migration	Does not consider space parameter in terms of consumed energy
Tarighi et al.	-Fuzzy decision making based VM migration scheduling algorithm	-VM migrations are not reduced -Energy saving is not achieved

[137]	-Migration decision using TOPSIS (Technique for Order Preference by Similarity to Ideal Solution approach)	
Tolia et al. [140]	-Short-term VM migration -Consolidates workloads & under-utilized servers put to sleep mode	Does not consider energy consumed by memory
Wood et al. [138] [118]	-Two gray-box and black-box approaches -VM live migration & diminishes hotspots	Number of VM migrations are not reduced to save energy
Ghribi et al. [135]	-Combines VM allocation and VM migration algorithms -Reduces the number of nodes -Saves energy	-Only reduces nodes -VM migrations not reduced -Energy consumed by memory is not considered
Marzolla et al. [139]	-Coarse-grained gossip based VM consolidation protocol -Migrates the VMs from smallest laden node to greatest laden node	-Energy saving is not achieved -No reflection of number of VM migrations
Goiri et al. [142]	-Multi-objective profit-oriented algorithm to place VMs -Performance in terms of SLA violations, energy efficiency and virtualization overheads	-Energy consumed by memory is not considered
Verma et al. [143] [144]	-Framework for VM placement algorithms -Considers energy & migration costs -Maximizes performance -Minimizes energy consumption	Does not consider space parameter in terms of consumed energy
Sekhar et al. [151]	-Greedy heuristics based energy efficient VM live migration technique -Curtails consumed energy	-Does not deal with CPU & memory utilizations -VM migrations & hosts not diminished
Murtazaev et al. [156]	-Server Consolidation technique -Reduces active nodes, VM migrations & consumed energy	-Energy saving parameters & values are implicit -Energy values are not computed
Jung et al.	-Framework for energy optimization -Uses live VM migration	-CPU & memory utilization are not considered

[152]	-Consolidates virtual servers -Switches-off the idle servers	-Does not lower hosts & VM migrations
Mehta & Neogi [145]	-ReCon tool to consolidate servers dynamically -Considers static and dynamic costs of physical servers & cost of VM migration -Considers past resource consumption data	Does not consider energy consumed by memory
Cardosa et al. [127]	-VM consolidation approach -Resource utilization is improved -Energy consumption is reduced	-Requires the knowledge of application priorities -Considers only CPU as a resource -Reduces only nodes not VM migrations
Beloglazov et al. [123]	-Energy-aware heuristics to allocate VMs dynamically & live migration of VMs -Minimizes energy consumption -Reduces number of used nodes -VMs applications' knowledge not required	-Does not include energy consumed by memory and network resources -Lowers used nodes only -Reduction of VM migrations is not considered
Bila et al. [153]	-A technique to migrate VMs partially -Reduces energy consumption	-Does not deal with CPU & memory utilization, Also VM migrations & hosts are not reduced
Mastroianni et al. [148]	-VM placement and consolidation based on Bernoulli trials -Considers energy & migration cost	-Emphasis on VM placement rather than migration & does not focus on number of VM migrations -Lowers used servers only
Vu et al. [150]	-VM placement algorithm -Enhances communication performance, CPU utilization & saves energy -Decreases overall VM traffic's cost	-Memory utilization not considered -VM migrations not reduced -Hosts are not saved
Xiaoli et al. [155]	-Energy-aware VM placement algorithm -Increases resource utilization -Enhances energy efficiency -Considers energy cost in migrations	-Does not deal with CPU & memory utilization -Does not lower hosts & VM migrations

Based on the investigation of the existing works, it can be inferred that, most of the techniques have focused on energy management largely through VM consolidation and VM placement. These techniques, if at all offer the energy optimization, do it at the processor level and do not consider it at the memory level. Apart from this, the mentioned techniques attempt to minimize the number of hosts , whereas there is no reflection of either the number of VM migrations or the attempt to reduce them.

With rising diversity and complexity of large-scale distributed computing services, there is a necessity to design more scalable, heterogeneous and sustainable computing techniques that can conjointly deal with the other issues such as heterogeneity and growing energy crisis as well. Thus, apart from the underlying infrastructure support (available through cloud computing in this case), it is important to explore and adopt new paradigms.

Currently, many researchers are focusing and implementing the biologically inspired computing as a preferable paradigm to handle these issues with proficiency and without the augmented complication. In spite of the several inherent challenges encountered while surviving in an enormous, dynamic, incredibly diverse, and highly complex environment, the biological organisms evolve, self-organize, self-repair, navigate, and flourish. This is possible with their local knowledge and without any centralized control [158] [159]. This prompted the research community to discover and learn lessons from the biological systems such as Ant Colony Optimization (ACO) [160] [78] [161] [162], Artificial Bee Colony (ABC) [163], Bacterial Foraging Optimization (BFO) [164], Particle swarm optimization (PSO) [165] [161] techniques etc.

Several Bio-inspired techniques that exploit the behavioural and social instincts of the biological creatures exist today. The following section discusses about these bio-inspired techniques in detail.

2.5 Bio-Inspired Techniques

Bio-inspired techniques are the methods which are highly efficient for solving combinatorial optimization problems. However, when the objective is to find feasible solutions of good quality in short execution times, as in the case of cloud resource scheduling, the inherent mechanisms of these methods can be exploited to increase the convergence of the method [166].

Genetic Algorithms (GA): GA, a famous stochastic optimization algorithm which uses biologically inspired techniques such as genetic inheritance, natural selection, mutation, and sexual reproduction (recombination, or crossover) [167], has been proposed by Holland et al [168]. It is a useful heuristic to discover a near optimal solution in large search spaces [169]. In GA, a point in search space is represented by a set of parameters known as genes. A set of genes is known as a string or a chromosome whereas a set of chromosomes is called population. A fitness function must be contrived for every problem to be solved. A fitness value is assigned to each chromosome to indicate its close association with the desired objective. The fitness function returns a single numerical fitness to determine the ability of the individual represented by a particular chromosome [170]. Another crucial attribute of GAs is reproduction where two individuals chosen from the population are allowed to mate to produce offspring, which will comprise the next generation. Having selected two parents, crossover and mutation mechanisms are used to combine their chromosomes. A small amount of random search is provided by mutation to ensure that no point in the search space has a zero probability of being examined. If the GA has been accurately carried out, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum. The GAs have been found to be very powerful in finding out a global minima [171] [172] and have been applied to many classification and performance tuning applications in the knowledge discovery domain of databases [167] [173].

The important advantages of GA are: (i) No analytical knowledge is required (ii) Easy to parallelize iii) No derivatives are required (iv) Works on a wide range of problems and has better global capability. The disadvantages of GA are: (i) Requires much more evolution functions than linearized methods (ii) No guaranty of convergence of local minima (iii) Converges to local optima or arbitrary point rather than the global optima of the problem (iv) Slow convergence rate and premature convergence and (v) Can not use the feed back of a system.

Memetic Algorithm (MA): MA, which is an extension of GA, is an evolutionary algorithm that can be applied on a local search process to refine solutions to hard problems. It is the subject of intense scientific research and has been successfully applied to a multitude of real-world problems ranging from the construction of optimal university exam timetables, to the prediction of protein structures and the optimal design of space-craft trajectories [174]. MA can handle complex objective functions and combines the advantages of local search with GA for optimization problems. It

can also be used for global search. It is based on a GA and is extended by a search technique to further improve individual fitness to keep up with the population diversity and reduce the likelihood of premature convergence. To improve its performance, MA requires a considerable amount of time and memory. It can be used only in non-linear continuous multi-objective combinatorial optimization problems.

Ant Colony Optimization (ACO): ACO, which is a meta-heuristic for finding near optimal solutions using a probabilistic technique, has been proposed by Marco Dorigo in 1992 [175]. It can be used for problems which belong to the NP class. ACO algorithms have been discovered by observing natural food-discovering behavior of real ants. Ants communicate indirectly using their environment. They deposit pheromone which is a chemical substance, that controls the behavior of ants when they encounter it. This method of communication is called stigmergy. Ants use probabilistic decision for their travel while searching for the food. They are more likely to choose paths with higher quantity of pheromone. When ants find food, they deposit pheromone on the return in order to induce other ants to follow the path until they reach the food source. Pheromone evaporation correspond to the natural evaporation of chemical substance deposited by ants. It is used to reduce the amount of pheromone over time for keeping only trails that are regularly used. This system aims to promote trails which lead to better solutions. When applied on combinatorial problems like BPP, artificial ants act as a multi-agent system and construct a complex solution based on indirect low-level communication [160] [78] [130]. The real power of ants resides in their colony brain. The self-organization of those individuals is very similar to the organization found in brain-like structures. Like neurons, ants use mainly chemical agents to communicate. One ant releases a molecule of pheromone that will influence the behavior of other ants [176]. Ant algorithms are non-deterministic and rely on heuristics to approximate to a sub-optimal solution in cases where the number of combinations is extremely huge and is impossible to calculate using a deterministic algorithm [160] [78].

Following advantages of ACO have been identified in [160] [78] [130]: (i) Versatile and can be applied to similar versions of the same problem (ii) Robust and can be applied with only minimal changes to other combinatorial optimization problems (iii) Can be used for static and dynamic combinatorial optimization problems (iv) Used for solving constrained discrete problems (v) Has powerful feedback capability to increase the speed of evolution of algorithm to make its convergence possible (vi) Guaranteed convergence (vii) Can run continuously and adapt to changes in real time. Some of the disadvantages of ACO are: (i) Convergence rate is slow (ii) Performs poorly for larger

city in TSP (iii) No centralized control to guide and provide good solutions (iv) Can only be applied to discrete problems and (v) Theoretical analysis is difficult.

Particle Swarm Optimization (PSO): PSO is one of the latest evolutionary optimization techniques inspired by nature and has been introduced by Kennedy and Elberhart [177] in 1995. It is a method to perform numerical optimization without the explicit knowledge of the problem gradient. It simulates the process of preying birds swarm. Due to its ability of global searching, it has been successfully applied to many areas. A flock of particles is randomly generated where position of each particle position represents a possible solution point in the problem space. The objective function to be optimized evaluates the fitness value of each particle. Each particle stores the coordinates of the best solution (gbest) and the current global best (pbest) solution.

The key advantages of PSO are: (i) Robust stochastic optimization based on the movement and intelligence of swarms (ii) No selection and crossover parameter like GA (iii) Easy to implement (iv) Few parameters to adjust (v) Computationally efficient and (vi) Efficient for global search algorithm Some of the disadvantages of PSO are: (i) Weak local search (ii) Slow convergence rate.

Bacterial Foraging Optimization (BFO): BFO algorithm, which is a population based numerical optimization algorithm based on foraging behavior of Escherichia coli bacteria, has been proposed by Passino [164]. In the foraging theory, the objective is to search and obtain high quality of nutrients to maximize energy intake per unit time (E/T). The bacterial foraging process consists of three main mechanisms: Chemotaxis, Swarming, Reproduction and Elimination-dispersal event. Chemotaxis is the process of simulating the movement of E.coli bacteria, which is carried in a flagella, through swimming and tumbling. The cell also repels a nearby cell in the sense that it consumes nearby nutrients and so it is not physically possible to have two cells at the same location. A bacterium in times of stress releases attractants to signal the bacteria to swarm together. After chemotaxis steps, a reproduction step is taken. Fitness value of bacteria is sorted in an ascending order. The least healthy bacteria eventually dies while each of the healthier bacteria (those yielding lower value of the objective function) asexually splits into two bacteria, which are then placed in the same location. This keeps the swarm size constant. Elimination event may occur due to sudden changes like a significant local rise of temperature or a part of them may move to other regions in the environment that will effect the behavior of bacteria heavily. The elimination and dispersal event destroys the performance of chemotaxis event but dispersal may place bacteria near good sources of food [178]. The main advantages of BFO are: (i) Easy

to implement, few parameters to adjust, computationally efficient (ii) More adaptive and can be easily applied to real world optimization problems and is efficient for global search (iii) Avoids the chance of premature convergence (iv) Robust and flexible to implement (v) Performance is high with respect to speed of convergence, quality of solution and rate of success.

Artificial Bee Colony Optimization (ABC): ABC algorithm is a swarm intelligence-based optimization algorithm, which has been introduced by Dervis Karaboga in 2005 [179] [180]. It simulates the foraging behavior of real bees and is used to optimize multidimensional functions. In this, bees are grouped as: employed bees (bees responsible for exploiting the nectar source and sending information about the quality of the food source sites to the onlooker bees), onlooker bees (bees waiting on the dance area to decide on a food source to exploit based on the information shared by the employed bees) and scout bees (bees carrying out random search to find a new food source). One half of the colony consists of the employed artificial bees and the other half includes the onlookers. There is only one employed bee for every food source i.e the number of employed bees is equal to the number of food sources around the hive. Initially, the foraging process begins by scout bees, i.e. all the food source positions are discovered by them. Upon their return to the hive from a foraging trip, they communicate the information regarding the food source (including its nectar amount, the direction in which it will be found and its distance from the hive) to onlookers by performing the waggle dance. Depending on this information, onlooker bees are recruited to exploit the food sources. If the scouts discover rich food sources then the scout bees are selected and classified as employed bee. After waggle dancing, these bees leave the hive to get nectar with onlooker bees. The number of onlooker bees assigned to fetch the nectar, depends on the overall quality of the nectar. Upon arrival to the food sources, the bees take a load of nectar and return to the hive. Here, they relinquish the nectar to an onlooker bee acting as a food-storer. In this way, the nectar of food sources are exploited by employed bees and onlooker bees, and this continual exploitation will ultimately cause them to become exhausted. The employed bee whose food source has been exhausted by the bees becomes a scout bee in search of further food sources once again. A possible solution and its associated quality(fitness) is represented by the position and the nectar amount of a food source respectively. This algorithm assumes a number of food sources and works through optimizing them. There are mainly four selection processes in ABC algorithm [181] [124] [78].

The main advantages of ABC are : (i) Simple to implement due to very few control

parameters (ii) Uses solutions with the higher fitness values than lower fitness values to produce trial solutions, leading to shorter time of search thereby balancing computational overhead and memory limit problems (It means that the ABC algorithm makes a decision based on a fitness value. Every time, the solution with the lower fitness value is replaced with a solution with higher fitness value. So, There is no need to examine all the solution candidates generated from the beginning to the end at the final step, and (11) High speed of convergence [181].

Firefly Optimization (FFO): FFO algorithm has been designed by Xin-She Yang in the late 2007 and 2008 at Cambridge University [165] [182] [183]. It is centred on the flashing features of fireflies and uses the subsequent three idealized procedures: (1) One firefly is attracted to the other fireflies irrespective of their sex as all fireflies are unisex, (2) The attractiveness is proportionate to the brightness, thus they both decrease as their distance increases and for any two flashing fireflies, the less brighter one will travel near the brighter one. If no firefly is brighter than a specific firefly, it moves arbitrarily and (3) The brightness of a firefly is regulated by the landscape of the objective function to be optimized.

The main advantages of FFO are: 1) Offers systematic partitioning and capability to handle multiple modes, (2) Computation time is less in possibility of finding the global optimized answer, (3) High speed of convergence which is due to the quality parameters that can be regulated, (4) A balanced and optimal solution is obtained by properly exploiting and exploring the problems search space, (5) Involves lesser number of function evaluations, (6) Its status can be changed from one optimization point to the other and (7) Random variables are used and the answers have the probable nature [182] [183] [184] [185].

Table 2.5 describes the comparison between the above discussed bio-inspired approaches with respect to their features. The comparison of the above described approaches depicts that ABC and FFO are better in comparison to the other approaches as both have high speed of convergence and have very few control parameters thereby leading to less computation time and high performance making them highly applicable in cloud environment.

Table 2.5: Comparison of Different Bio-inspired Approaches

Techn-ique	**No. of Para-meters**	**Convergence**	**Services**	**Search Space**	**Optimization Problems**
GA	More	Not guaranteed, premature convergence	No need of analytical knowledge,easy to run & implement	Global	Multi-objective optimization problems
MA	More	Guaranteed convergence, Less chance of premature convergence	Flexible	Global	Complex objective functions, non-linear multi-objective combinatorial optimization problems
ACO	Less	Guaranteed convergence, Avoids premature convergence	Versatile & robust	Global	Static and dynamic combinatorial optimization problems
PSO	Less	Slow rate of convergence,Less chance of premature convergence	Robust	Global	Stochastic optimization problems
BFO	Less	Better rate of convergence, Avoids premature convergence	Flexible & Robust	Global	Real-world optimization problems
ABC	Few	High speed of convergence	Robust, Uses solutions with higher fitness values than lower fitness values	Global	Multidimensional functions
FFO	Few	High speed of convergence	Robust, Offers system-atic partitioning & capability to handle multiple modes	Global	Higher-dimentional optimization problems

After analysing various bio-inspired techniques in this section, the next section focuses on the merits of bio-inspired approaches for energy conservation.

2.6 Merits of Bio-inspired approaches for Energy Conservation

Accomplishing energy optimization through optimum resource utilization and VM migration, by regulating workload on individual nodes, is an NP (Nondeterministic Polynomial) - hard problem [186] [187], and thus, optimal algorithms are not scalable (due to exponential cost of finding an optimal solution) which is compulsory in large scale systems such as a cloud. The solution is to compute near optimal solutions using a heuristic by applying greedy or bio-inspired algorithms [188]. To find an optimal solution, local heuristics may not be adequate, therefore, meta-heuristic approaches are suitable to efficiently crack these types of problems. Meta-heuristic is a repetitive primary procedure to guide and amend the jobs of secondary heuristics to yield high-grade results [189].

Currently, many researchers are focusing and implementing biologically inspired computing as a preferable paradigm to handle heterogeneity and growing energy crisis with proficiency and without augmented complication. They have been working towards using the role of biological systems to manage the energy problems. Several techniques as presented in Table 2.6, have been proposed in cloud computing to achieve energy efficiency, by utilizing the capabilities of these bio-inspired systems.

The variation of PSO, a Simplified Swarm Optimization (SSO) method has been proposed by Bergmann et al. [190]. It is based on distinct standards for reducing energy consumption in cloud computing systems. The energy consumption is optimized without degrading the performance, by using a DVS technique. The SSO method minimizes the makespan and power usage at its prime level.

Wang et al. [191] have proposed a multi-task scheduling technique using GA to exploit the vast processing capacity of the Google network. The technique uses some training and interpreting techniques for network entities and then constructs a suitable energy function based on the robustness of each entity. The main aim of this approach is to optimize the CPU utilization, as the energy consumed by each task depends on the CPU utilization of the server making it difficult to handle.

Moghaddam et al. [192] have proposed a technique that is based on Low Carbon Virtual Private Cloud (LCVPC). This technique reduces the carbon emissions of remote

data centers operating in multiple areas, but interconnected through personal Wide Area Networks(WAN) or the Internet. An intelligent live VM migration within a WAN has been used. The first calculates The carbon footprint as well as the power consumed by the entire network is calculated by the VPC manager before applying a modified GA. The carbon emission levels are further optimized by using dynamic VM consolidation.

Nguyen et al. [193] have designed another GA-based technique, namely, the Genetic Algorithm for Power-Aware (GAPA) scheduling for resource allocation that finds an optimal VM allocation strategy, to solve the static VM allocation problem.

A parallel bi-objective hybrid Genetic Algorithm (GA) has been developed by Kessaci et al. [194]. This algorithm uses a hybrid scheduling algorithm based on Energy-Conscious Scheduling (ECS) heuristic and the multipurpose parallel GA to fulfill the dual objectives of task completion time and energy consumption. The scheduling process minimizes the makespan and keeps the energy consumption level to low by allocating 'k' tasks to 'n' processors.

A technique to provide an energy-efficient resource allocation mechanism has been proposed by Chimakurthi and Madhukumar [195]. This technique uses an adaptive allocation mechanism to allocate the resources to various applications according to their QoS requirements. It is based on ACO technique , and to reduce the power consumed by the data center resources, ant agents are modeled as changing server loads.

Feller et al. [130] have presented a bio-inspired workload consolidation technique that is based on the ACO meta-heuristic. The proposed algorithm makes dynamic decisions to place workloads and overcomes the workload consolidation problem of single resource relevance. The difficulty in workload assignment is modeled as a multidimensional bin-packing problem.

Table 2.6: Comparison of Bio-inspired Computing-based Approaches [33]

Techniques	**Basis**	**Benefits**	**Drawbacks**
Simplified Swarm Optimization [190]	DVS and swarm intelligence	Tasks proposed for other communal constituents can be designed	Not practicable for non-time-sensitive systems
Scheduling algorithm	Grounded on Googles	A load balancing	-No assurance of QoS

for Energy-efficient multitasking [191]	enormous statistics handling web	method is implemented	& cost minimization -Extended computational time
Modified GA [192]	VMs are migrated perceptively & with dynamism united with an empirical approach	Compendious cost function appropriate for the whole system is offered	-Diminishes carbon footmarks -Overlooks performance metrics
Genetic Algorithm for Power-Aware [193]	Functions in private cloud	Curtails energy consumption	Computational time is extended
Parallel bi-objective amalgam GA [194]	Multi-objective GA and ECS heuristics based	Optimization based on deadline is provided	Cost enhances due to the consideration of multi-objectives
An energy-effective, agent-based solution [195]	Jobs destined to CPU and memory are assigned with dynamism	-Enhanced throughput & response time -Observes SLAs	Load forecast is absent
Workload consolidation approach based on energy-aware ACO [130]	Ant Colony Optimization meta-heuristic based	-Needs less nodes -Improved resource utilization	-Great enslavement on workload & resource facts -Cannot deal with dynamic nature of tasks

Based on the analysis of various bio-inspired techniques, it can be inferred that these techniques offer near-optimal solution to realize energy efficiency. Thereby, the proposed approaches of this work, benefit from the merits of the existing bio-inspired approaches. This work has chosen ABC & FFO for designing the ABC-based ERU & FFO-based EVMM. The reasons for choosing ABC optimization technique over other social behavior inspired techniques are : (1) it is simple to implement as there are very few control parameters, (2) it uses solutions with the higher fitness values than lower fitness values to produce trial solutions, leading to shorter time of search and (3) it has high speed of convergence [181], whereas the criterion for choosing the biological

behaviour of the firefly insects is its faster convergence speed and global optimization attainment [182].

This section presented the merits of bio-inspired approaches for energy efficiency. The next section formulates the problem statement for this research work, based on the findings of the available literature.

2.7 Problem Statement & Research Objectives

As per the literature survey, one of the main issues in cloud computing is Load balancing. In cloud computing, load balancing is required to redistribute the workload among nodes to avoid a situation where some nodes are heavily loaded while others are idle or doing little work. Load balancing helps to achieve high user satisfaction and resource utilization ratio. With proper load balancing, hot-spots can be prevented and resource utility levels can be improved, which can further reduce energy consumption thereby, reducing carbon emission and cooling requirements of the cloud data centers, which is a dire need of cloud computing.

Therefore, there is a need to develop a near optimal load balancing technique that can improve the performance of cloud computing by balancing the workload across all the nodes in the cloud along with maximum resource utilization, in turn reducing energy consumption as well as carbon emission and cooling requirements to an extent which will help to achieve green computing. In this work, Energy-aware Load Balancing (ELB) techniques are developed that are based on the optimized resource utilization and VM migration to achieve energy efficiency. The objectives of this research work are:

1. To analyze existing load balancing techniques in cloud computing.
2. To design and develop energy-aware load balancing techniques for an optimal balance between performance and energy consumption.
3. To test and validate the proposed load balancing techniques in cloud environment(s).

2.8 Conclusion

This chapter discussed the research work accomplished in the area of load balancing, resource utilization, VM migration and bio-inspired techniques. None of the discussed

techniques have used CPU and memory utilization, performance and energy consumption together.

The next chapter presents the Energy-aware Resource Utilization model to manage and enhance the resource utilization.

Chapter 3

Proposed Energy-aware Resource Utilization Model

The previous chapter surveyed the research work accomplished in the area of resource utilization,virtual machine migration and load balancing techniques along with the identification of the need of designing energy-aware load balancing techniques.

This research work proposes the design of Energy-aware Load Balancing Techniques. Optimum resource utilization and virtual machine migration contribute significantly towards reducing energy consumption in cloud data centers as reported in the literature survey. Therefore, the proposed load balancing techniques presented in this thesis are based on these two aspects. This chapter discusses the Energy-aware Resource Utilization (ERU) Model that is used in efficient load balancing. The ERU model is based on the Artificial Bee Colony (ABC) optimization technique and efficiently manages the cloud resources to enhance their utilization. It also maximizes the energy-efficiency through the best usage of the resources and without degrading the system performance.

This chapter initially presents the requirements, architecture and components of the proposed ERU model followed by the system model and the fitness function. Further, it describes the ABC optimization technique which is the basis of the proposed ERU model, pursuing the detailed design, algorithm and the flowchart of ERU. Later, it demonstrates the implementation details of ERU by using the CloudSim Simulator.

3.1 Energy-aware Resource Utilization (ERU) Model

The proposed ERU model provides the facility to manage the resources efficiently and the aim of this model is to reduce energy consumption of clouds, besides preserving the system's performance. The resources considered are the CPU and the memory, as these are the prime contenders to consume energy in the cloud data center [196] [33] [123] [157] [41]. The ERU model gives access to the required resources according to the users' requirements. This is done in such a way so that the maximum utilization of the resources is achieved, thereby enhancing performance and energy-efficiency of the data center. The key features of this model are:

- Managing cloud resources with respect to energy consumption to make energy-aware scheduling decisions without degrading the performance
- Enhancing resource utilization to reduce energy consumption of clouds
- Keeping idle physical nodes in a sleep mode to save energy
- Using energy consumption calculator to monitor energy consumed by the cloud resources
- Enhancing performance and energy-efficiency of cloud data centers

In this approach, a virtualized environment is considered where all the nodes have the same configuration and have heterogeneous VMs that are created at the run-time according to the workload's resource requirements. The heterogeneous workloads are accepted from the cloud user. It is important to carefully consolidate variable workloads in order to avoid contention of resources which may further cause performance degradation and energy wastage. All the nodes are kept in an energy-saving state and brought to the ON-state, only when required. Various terms used in this model are as given in Table 3.1. Figure 3.1 shows the ERU model and various modules of this model are as follows :

- Cloud Portal - It helps the cloud user to submit a workload to the cloud and to get the resources to execute this workload from the cloud. Each cloud user submits a workload (set of jobs) to the cloud and gets the resources to execute the workload from the cloud through the cloud portal. The user enters the required number of CPUs, required amount of memory and the type of the workload (whether CPU-intensive or memory-intensive).

Table 3.1: Description of Various Terms

Terms Used	Description
VM	Virtual Machine is an emulation of a computer system
Server/Node/Host	A rack-mountable hardware or a computer system where VMs are running
Resource	CPU/Memory
Virtualized Environment	Nodes with same configuration but heterogeneous VMs created at run-time
Heterogeneous Workloads	CPU-intensive/Memory-intensive
Cluster	A group of homogeneous nodes

* Server, Node and Host are used interchangeably

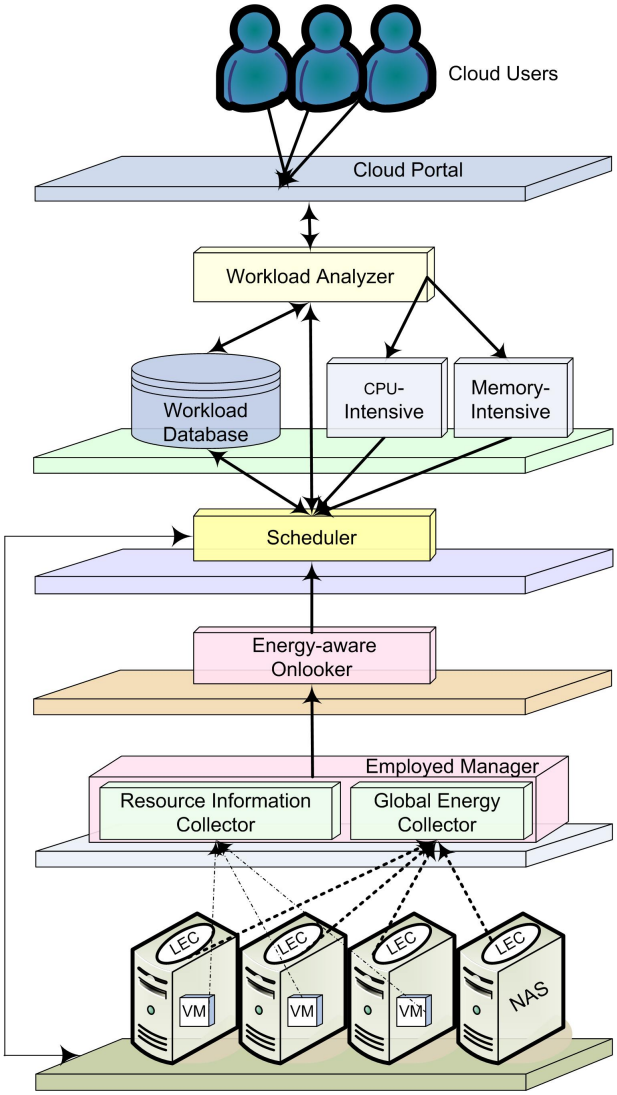


Figure 3.1: Energy-aware Resource Utilization Model

- Workload Analyzer (WA) - It analyzes the workload before sending it to the scheduler. The classification of the CPU and memory intensive workloads have been done to carefully consolidate them on the nodes to avoid the contention of the resources and the conflicts between the processor and the memory utilizations.
- Workload DataBase (WDB) - It is a repository to store the past resource utilization and the energy consumption data of the workloads which have already been entered into the system and have used the system resources. On an average, similar workloads entered into the system after about 5 minutes.
- Scheduler - It schedules the workload to the required resources. It is named as ABC-Scheduler after the name of the ABC technique which is the basis of ERU model.
- Energy-aware Onlooker (EO) - It decides about the most energy-aware node meeting the resource requirements for the workload and provides this information to the scheduler.
- Employed Manager (EM) - It provides the resource-related information and the energy-related information of all the nodes to the EO module. It is further comprised of the following two sub-modules :
  - Resource Information Collector (RIC) - It gathers the information about all the resources in the cluster.
  - Global Energy Collector (GEC) - It collects the information about the energy consumption of all the nodes in the cluster.
- Local Energy Calculator (LEC) - It is deployed on each node and calculates the energy consumed by the respective node. This information is then provided to GEC in EM module.
- Network Attached Storage (NAS) - In the data center, one of the system nodes is used as NAS, which is a node that provides a central storage system for all the files in one place. It allows more hard-disk storage space to be added to a network.

Working of ERU Model :

Upon initial submission of the workload, as its resource utilization and energy consumption data is not available in WDB, resources are allocated to it according to the information provided by the user. Once the workload gets processed, its resource

utilization and energy consumption data becomes available and is used for future reference. Next time, when the same workload arrives, this stored data is utilized for making a decision about the resource allocation. i.e. when the user submits a job, it is checked for its past resource utilization and energy consumption data. If this data is not present, the workload is allocated to the resources according to user's requirements and its data regarding the used resources and the consumed energy is stored in WDB, otherwise the present data is used for the resource allocation. After this phase, the workload gets factorized as CPU-intensive or memory-intensive. This factorization is done to avoid a situation of allocating the same type of workload to the same node beyond its threshold. This situation may lead to the contention of that particular resource, resulting in the poor performance of the system.

Once the workload has been submitted to the portal, it is given to WA module. WA queries WDB module about this workload. WDB checks for the entry related to the query submitted by WA. If it finds a match, it returns the related data to WA otherwise it returns a "No match found" message. Depending on the data received from WDB, WA either preprocesses the workload and sends it to the scheduler or directly sends the workload to the scheduler. The old workload i.e. the workload with the existing past resource utilization and energy consumption data, is factorized as CPU-intensive or memory-intensive before sending it to the scheduler and the new workload i.e. the workload without the past resource utilization and energy consumption data, is sent directly to the scheduler.

Now, the scheduler makes an energy-aware scheduling decision and maps the workloads to their required resources in co-ordination with EO as shown in Figure 3.2. The scheduler schedules the workload received from WA directly or indirectly, to the ingredient resources according to the information collected from EO module.

The EO module collects the resource-related information from RIC module of EM and the energy-related information from GEC module of EM. After gathering this information, it makes a decision about the most energy-aware node meeting the resource requirements for the workload.

EO first checks for the available resources of one particular node. If the resources of this node are not sufficient to meet the requirements of the workload, it looks for the next node. If the available resources fulfil the required condition, it further checks whether the required resources are exceeding the threshold values of the resources, set for that node. If they exceed, it checks the next node otherwise it further checks whether the execution of this workload rises the energy consumption of that node above

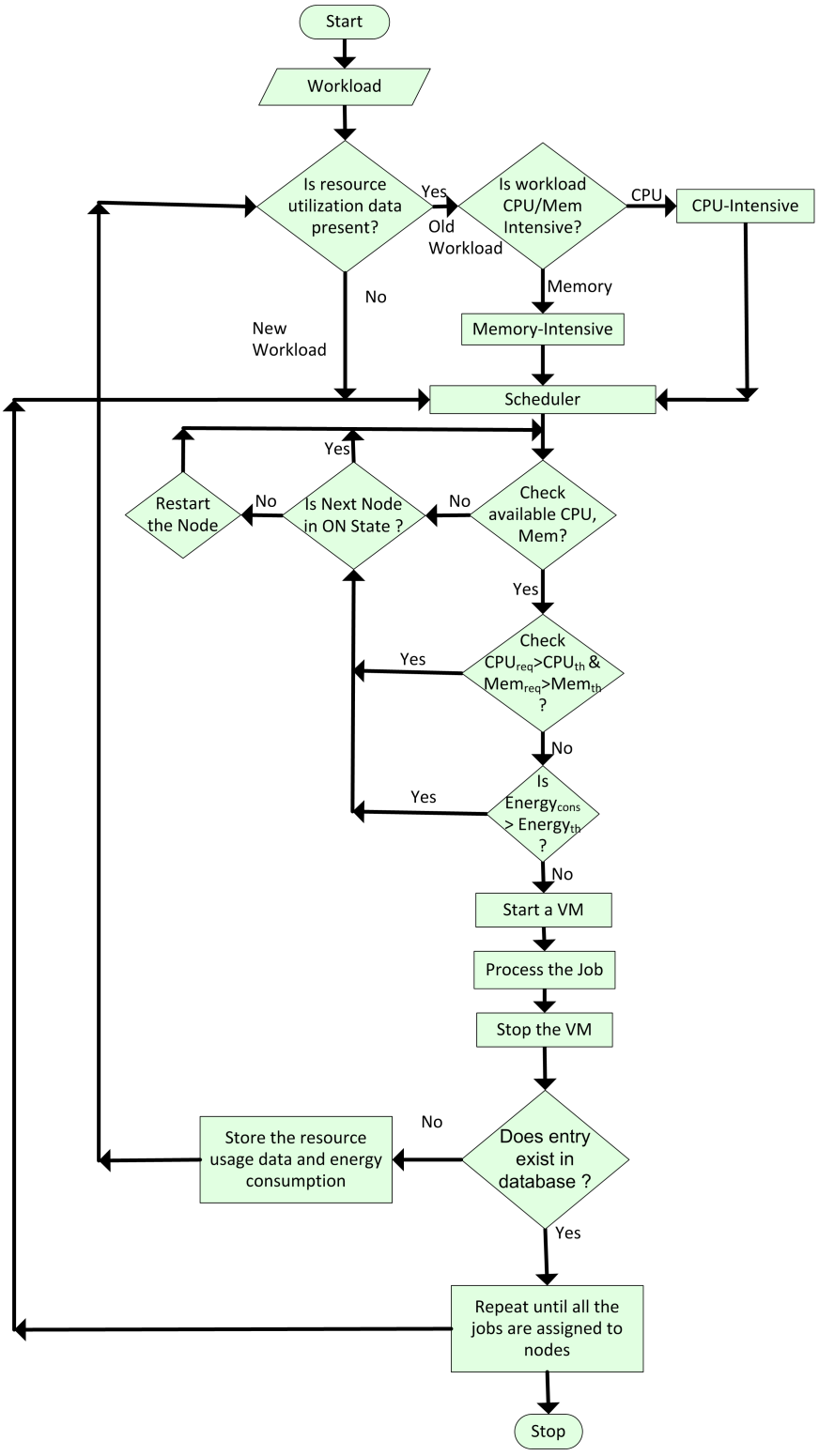


Figure 3.2: Flowchart for Energy-aware Resource Utilization Model

a set threshold. If the energy consumption goes beyond the threshold, the EO searches for the next node otherwise, a VM, fulling the resource requirements of the workload, is started on that node to process the workload and after the completion of the job, the VM is stopped for saving energy. After this, a database entry related to the past resource utilization data of this processed workload is searched. If no such entry exists, the resource utilization and energy consumption data of this workload is saved in the WDB otherwise it exits without saving anything. When all the nodes that are in ON-state, are searched and no node fulfils the requirements of the incoming workload, a new node which is in energy-saving mode, is turned ON for processing that workload. This is repeated for each and every workload entering into the system.

The information about the most energy-aware node is then provided to the scheduler to make an appropriate energy-aware scheduling decision. According to the decision made by the scheduler, the workload is allocated to the optimal resources. After the completion of workload execution, the data regarding the resources used and energy consumed is stored in the WDB. Thus, this model exhibits how energy-aware based resource scheduling can be done in the cloud environment.

To find the best node to a corresponding job is a tedious task and the problem of finding the best node for a job, according to the user's requirements is a combinatorial optimization problem. The main goal of the scheduler in ERU model is to schedule workloads to the resources of nodes effectively and efficiently. Assigning jobs to appropriate nodes is an NP-hard problem [197], and thus, optimal algorithms are not scalable (due to exponential cost of finding an optimal solution) which is compulsory in large scale systems such as a cloud. The solution is to compute near optimal solutions using a heuristic by applying greedy or bio-inspired algorithms [188]. Thus, Artificial Bee Colony (ABC) optimization technique has been used as as the basis of this model. The proposed approach is used to assign jobs to the nodes to save energy by efficiently utilizing the resources and preserve the performance by minimizing the execution time of the jobs. The optimization problem optimizes energy as well as execution time through a fitness function.

For implementation of ABC-based ERU, system equations and fitness function are required that form the basis of the system behavior. In the following subsection, various system equations have been proposed followed by the fitness function for the implementation of the proposed technique.

3.1.1 System Model

In this work, the scheduling problem has been considered from cloud provider's point of view as he wants to minimize the Execution Time (ET) of the jobs and Energy Consumption (EC) of the nodes. In this problem, the minimization of the consumed energy in the data center is considered. Smaller values of EC indicate that the scheduler is planning the jobs in an energy-aware manner. ET is another optimization criterion, which refers to the execution time of the jobs, running in VMs, on the nodes. The problem has been mathematically formalized to get a near-optimal solution.

To consider this problem, a set $\{r_1, r_2, ..., r_M\}$ of nodes such that $R = \{r_i | 1 \leq i \leq M\}$ is the collection of M nodes; a set $\{v_1, v_2, ..., v_N\}$ of VMs such that $VM = \{v_j | 1 \leq j \leq N\}$ is the collection of N VMs; and a set $\{job_1, job_2, ..., job_L\}$ of jobs to be run in VMs such that $JOB = \{job_k | 1 \leq k \leq L\}$ is the collection of L jobs, have been taken. Let c_i be the overall CPU capacity in MIPS and m_i be the memory capacity in MB of a node r_i. Let vc_j and vm_j be the CPU and the memory capacities of a VM v_j.

For the problem formulation, the following constraints are taken into consideration:

1. A fully virtualized cluster is considered that has a pool of nodes of the same capacity with heterogeneous VMs running on them.

2. The jobs run in VMs, which are created dynamically according to the jobs' requirements.

3. Jobs are independent and need to be allocated across the node-pool.

4. Each node is characterized by its processor and memory utilizations.

5. The CPU and memory utilization of each node needs to be bound by upper threshold values to prevent them from reaching 100% utilization which can lead to performance degradation. These have to be chosen very carefully as very high values degrade the performance and very low values increase the energy consumption.

6. A threshold for energy consumption has to be set for each node in order to allocate workloads below this value to increase the overall energy-efficiency of the cloud.

7. Each user submits a workload. Workload is a set of jobs running in VMs. Workloads are submitted according to the specific requirements of entering its type, i.e. whether its CPU-intensive or memory-intensive.

At any given time, for i^{th} node , the CPU utilization (RPU_i) and the memory utilization (RMU_i) can be given as :

$$RPU_i = \sum_{j=1}^{n_i} \sum_{k=1}^{l_i} rpu_{ijk} \tag{3.1}$$

and

$$RMU_i = \sum_{j=1}^{n_i} \sum_{k=1}^{l_i} rmu_{ijk} \tag{3.2}$$

where n_i is the number of VMs running on the i^{th} node and l_i is the number of jobs assigned to n_i VMs. rpu_{ijk} and rmu_{ijk} are the CPU and the memory utilizations of k jobs running in j VMs on the i^{th} node respectively. As, ABC optimization is the basis of the proposed ERU model, therefore, the CPU and the memory utilizations need to be normalized. The normalized CPU utilization can be given as :

$$RPU_i = \frac{RPU_i - RPU_{min}}{RPU_{max} - RPU_{min}} \tag{3.3}$$

where RPU_{max} is the maximum and RPU_{min} is the minimum CPU utilization from all the nodes in the system.

Similarly, the normalized memory utilization can be given as :

$$RMU_i = \frac{RMU_i - RMU_{min}}{RMU_{max} - RMU_{min}} \tag{3.4}$$

where RMU_{max} is the maximum and RMU_{min} is the minimum memory utilization from all the nodes in the system.

Therefore, the power consumption, PC_i, of the i^{th} node, in terms of its respective CPU and memory utilizations, can be given as :

$$PC_i = RPU_i + RMU_i \tag{3.5}$$

And the Energy consumption of a node can be defined as the power consumption of that node during $'t'$ units of time. Therefore, Energy Consumption (EC_i) of the i^{th} node is calculated as :

$$EC_i = PC_i \times t \tag{3.6}$$

where PC_i is the power consumption of the i^{th} node during $'t'$ units of time. Hence the total energy consumption (EC) of the system is :

$$EC = EC_1 + EC_2 + ... + EC_M \tag{3.7}$$

where M is the total number of nodes. Therefore

$$EC = \sum_{i=1}^{M} EC_i \tag{3.8}$$

ET_i is the execution time of the jobs running in VMs on the i^{th} node and is defined as :

$$ET_i = \sum_{j=1}^{n_i} \sum_{k=1}^{l_i} ET_{ijk} \tag{3.9}$$

where ET_{ijk} is the execution time of k jobs running in j VMs on the i^{th} node.

Hence the total execution time (ET) is :

$$ET = \sum_{i=1}^{M} ET_i \tag{3.10}$$

3.1.2 Fitness function

The main goal, in this work, is to minimize the energy consumption and execution time by efficiently utilizing the resources of the nodes. The fitness value is thus calculated as :

$$fit_m = \theta(EC) + \delta(ET) \tag{3.11}$$

$$EC = min(EC(r_i, job_k)) for 1 \leq i \leq M, 1 \leq k \leq L \tag{3.12}$$

$$ET = min(ET(r_i, job_k)) for 1 \leq i \leq M, 1 \leq k \leq L \tag{3.13}$$

where $0 \leq \theta < 1$ and $0 \leq \delta < 1$ are the weights to prioritize the components of the fitness function. $EC(r_i, job_k)$ is the energy consumption of the job job_k on the the node r_i and $ET(r_i, job_k)$ is the execution time of the job job_k on the the node r_i.

Fitness function is subjected to the constraints :

$$\frac{\sum_{j=1}^{n_i} vc_{ij}}{c_i} \leq CPU_{th} \tag{3.14}$$

i.e. the ratio of the aggregated CPU capacity of VMs on the i^{th} node to the CPU capacity of the i^{th} node must not exceed the CPU threshold value, CPU_{th} of that i^{th} node,

$$\frac{\sum_{j=1}^{n_i} vm_{ij}}{m_i} \leq MEM_{th} \tag{3.15}$$

i.e. the ratio of the aggregated memory capacity of VMs on the i^{th} node to the memory capacity of the i^{th} node must not exceed the memory threshold value, MEM_{th} of that i^{th} node and

$$EC_i \leq Energy_{th} \tag{3.16}$$

i.e. the total energy consumption of the i^{th} node should not exceed its energy threshold value, $Energy_{th}$.

After an overview of the key characteristics, various modules, the data flow representation & the working of the proposed ERU model, this section presented the system model and the fitness function used in the model. The following section describes the Artificial Bee Colony (ABC) optimization technique which is the basis of our proposed ERU model.

3.2 Artificial Bee Colony (ABC) Optimization Technique

ABC algorithm is a swarm intelligence-based optimization algorithm, which was introduced by Dervis Karaboga in 2005 [163] [198]. In ABC, the colony of artificial bees contains three groups of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly. Both onlookers and scouts are also called unemployed bees. Initially, all food source positions are discovered by scout bees. Thereafter, the nectar of food sources are exploited by employed bees and

onlooker bees, and this continual exploitation will ultimately cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of further food sources once again. In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of employed bees is equal to the number of food sources (solutions) since each employed bee is associated with one and only one food source. Three control parameters used in ABC are SN (no. of food sources that is equal to employed/onlooker bees), the value of limit (the predetermined no. of cycles for which a position undergoes an improvement phase)and the MCN (Maximum Cycle Number) for which the population of the positions (solutions) is subjected to repeated cycles of the search processes of the employed bees, the onlooker bees and scout bees) [179] [180] [181].

The general algorithm of the ABC optimization technique is :

1. Initialization Phase
2. REPEAT
   - Employed Bees Phase
   - Onlooker Bees Phase
   - Scout Bees Phase
   - Memorize the best solution achieved so far

   UNTIL(Cycle=Maximum Cycle Number)

Four selection processes of ABC algorithm [179] [180] [181] are as follows :

(1) A **local selection process** carried out by the artificial employed bees and the onlooker bees in a region to produce a candidate food position from the old one by

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.17}$$

where $k \in \{1, 2, ..., SN\}$ & $j \in \{1, 2, ..., D\}$ are randomly chosen indexes. D is the number of optimization parameters and ϕ_{ij} is a random number between $[-1, 1]$.

(2) A **greedy selection process** carried out by all bees is

if *New(nectar amount) >Old(nectar amount)* **then**
    Memorize the new position

else
    Keep the old position in her memory

(3) A **global selection process** used by the artificial onlooker bees for discovering a food source depending on its associated probability value is given by

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3.18}$$

where fit_i is the fitness value of the solution i and SN is the no. of food sources.

(4) A **random selection process** carried out by scouts. A new random solution to replace the abandoned source x_i is produced by

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j) \tag{3.19}$$

where $rand(0,1)$ is a random number within $[0,1]$ based on a normal distribution and x_{min}^j , x_{max}^j are the lower and the upper boundaries of the j^{th} dimension, respectively.

This section described the ABC optimization technique, which is the basis of the proposed ERU model, in detail. The next section gives the detailed design, algorithm and the flowchart of the proposed ABC-based ERU.

3.3 ABC-based ERU

Energy-aware Resource Utilization (ERU) provides an optimal solution to find the best job to particular node pair.

Pseudo code of ABC-based ERU :

In this section, the pseudo code of ABC-based ERU has been presented where the input is a set of jobs and a set of nodes. Many parameters have been initialized as shown in Table 3.2. For the proposed ERU, • Scheduler is the hive for the bees, • Cluster is the region of ABC search space, • Nodes are the food sources and • CPU, memory and energy values of a node are considered the nectar of that node.

The pseudocode of ABC-based ERU is given in Algorithm 1. First of all, the workloads are analyzed and factorized according to the past data stored in workload data base. Then, the ABC-Scheduler is called to schedule the workloads to their required resources. The ABC-Scheduler is a module which makes use of ABC optimization technique to optimally allocate the resources to the workloads. It sends the scout bee

Table 3.2: Initialization Parameters

Parameter	Comment
(JOB)	Set of jobs
(R)	Set of nodes
$(PREU)$	Past Resource and Energy Utilization
(M)	Number of nodes
(MCN)	Maximum Cycle Number for each search process
(W_Type)	Workload type where CW is for CPU-intensive Workload and MW is for Memory-intensive Workload
$(Avail\_CPU, Avail\_Mem)$	Available CPU and Memory with every node
(CPU_{req}, Mem_{req})	Required CPU and Memory by each workload
$(Energy_{req})$	Energy required by each workload
(CPU_{th})	CPU threshold for each node
(Mem_{th})	Memory threshold for each node
$(Energy_{th})$	Energy threshold for each node

agents and employed bee agents to search the nodes for their resources. Employed bee agents share this information with the onlooker bee agents. Depending on the resources and the energy consumption values of the nodes and using a probability based selection process (Equation 3.18), the onlooker bee agents select the best energy-aware nodes. Once, all the nodes get exhausted, scout bee agents are sent to explore more nodes in the search space. The process has been repeated for a maximum cycle number with each iteration producing an improved solution (position of the node). The scheduler ends the process when all the jobs have been allocated to the best resources. The detailed description of each function of the scheduler is as follows :

Initialization_Phase()

All the vectors of the population of nodes, $(\vec{y_m})$s, are initialized $(m = 1...M, M :$ population size) by scout bee agents and control parameters M and MCN are set. Since each node, $(\vec{y_m})$, is a solution vector to the optimization problem, each $(\vec{y_m})$ vector holds n variables, $(y_{mi}, i = 1...n)$, which are to be optimized so as to minimize the objective function. Equation 3.19 is used for the initialization purposes as :

$$y_{mi} = l_i + rand(0, 1)(u_i - l_i) \tag{3.20}$$

where (l_i) and (u_i) are the lower and the upper bound of the parameter (y_{mi}), respectively.

Employed_Bees_Phase()

1. *Produce_New_Solution()* : Employed bee agents search for new nodes $(\vec{v_m})$ having

Algorithm 1: ABC-based ERU

```
Data: set of jobs, set of nodes
Result: Finding the best job-node pair
begin
    JOB ← set of jobs
    R ← set of nodes
    PREU ← boolean variable  // 1- If past resource and energy utilization data is present in WDB
    C = 1                                                  // Number of cycles for improvement phase
    InputMCN, M, W_Type, Avail_CPU, Avail_Mem, CPU_req, Mem_req, Energy_req, CPU_th, Mem_th,
    Energy_th
    for (each job ∈ JOB) do
        if (PREU = 0) then
            // If past resource and energy utilization data is not present, send jobs directly
            to scheduler
            ABC_Scheduler()
        else
                                          // Factorize the workload as cpu-intensive or memory-intensive
            if (W_Type = CW) then
                Assign_to_CPU()
            else
                Assign_to_Memory()
            ABC_Scheduler()
        Repeat until all the jobs are assigned to nodes
                                                                        // Definition of ABC_Scheduler
    ABC_Scheduler()
    begin
        Initialization_phase()
        begin
            (y⃗_m)s ← population of all possible solutions      // All possible solutions are initialized
            y_mi = l_i + rand(0,1)(u_i − l_i)
        while (C < MCN) do
            Employed_Bees_Phase()
            begin
                υ_mi = y_mi + φ_mi(y_mi − y_ki)                                  // Produce New Solution
                                                                // Memorize the Better Solution
                if (υ⃗_m > y⃗_m) then
                    memorize (υ⃗_m)
                else
                    keep (y⃗_m)
            Onlooker_Bees_Phase()
            begin
                if (CPU_req <= Avail_CPU) && (Mem_req <= Avail_Mem) then
                    if (CPU_req < CPU_th ) && (Mem_req < Mem_th ) then
                        if (Energy_req < Energy_th) then
                            p_mi = fit_mi(y⃗_m) / Σ_{m=1}^{M} fit_m(y⃗_m)          // Calculate Probability
                            υ_mi = y_mi + φ_mi(y_mi − y_ki)                      // Produce New Solution
                                                                // Memorize the Better Solution
                            if (υ⃗_m > y⃗_m) then
                                memorize (υ⃗_m)
                            else
                                keep (y⃗_m)
            Scout_Bees_Phase()
            begin
                y_mi = l_i + rand(0,1)(u_i − l_i)                    // Determine Abandon Solution
            Memorize Best Solution
            Increment C by 1
        Output the best solution achieved
```

more resources within the neighborhood of the node ($\vec{x_m}$) in their memory. They find a neighbor node and then evaluate its profitability (fitness). A neighbor node ($\vec{\upsilon_m}$) is determined using Equation 3.17 as :

$$\upsilon_{mi} = y_{mi} + \phi_{mi}(y_{mi} - y_{ki}) \tag{3.21}$$

where ($\vec{y_k}$) is a randomly selected node, (i) is a randomly chosen parameter index and (ϕ_{mi}) is a random number within the range $[-1, 1]$.

2. *Greedy_Selection_Process()* : After producing the new node ($\vec{\upsilon_m}$), its fitness is calculated and a greedy selection is applied between ($\vec{\upsilon_m}$) and ($\vec{y_m}$).

Onlooker_Bees_Phase()

1. *Calculate_Prob()* : Unemployed bee agents consist of two groups of agents - onlooker bee agents and scout agents. Employed bee agents share their node information with onlooker bee agents waiting with the scheduler and then onlooker bee agents probabilistically choose their nodes depending on this information. An onlooker bee agent chooses a node depending on the probability values calculated using the fitness values provided by employed bee agents.

   The probability value (p_m) with which ($\vec{y_m}$) is chosen by an onlooker bee agent is calculated by using the expression given in Equation 3.18 as :

$$p_{mi} = \frac{fit_{mi}(\vec{y_m})}{\sum_{m=1}^{M} fit_m(\vec{y_m})} \tag{3.22}$$

2. *Produce_New_Solution()* : After a node ($\vec{y_m}$) for an onlooker bee agent is probabilistically chosen, a neighborhood node ($\vec{\upsilon_m}$) is determined by using Equation 3.21, and its fitness value is computed using Equation 3.11.

3. *Greedy_Selection_Process()* : As in the employed bees phase, a greedy selection is applied between ($\vec{\upsilon_m}$) and ($\vec{y_m}$). Hence, more onlooker bee agents are recruited to richer sources and positive feedback behavior appears.

Scout_Bees_Phase()

Determine_Abandon_Solution() : The unemployed bee agents who choose their nodes randomly are called scout bee agents. Employed bee agents whose solutions cannot

be improved through a predetermined number of trials, called limit or abandonment criteria herein, become scout agents and their solutions are abandoned. Then, the converted scout bee agents start to search for new solutions, randomly. For instance, if solution ($\vec{y_m}$) has been abandoned, the new solution discovered by the scout bee agent who was the employed bee agent of ($\vec{y_m}$) can be defined by Equation 3.19. Hence those nodes which are initially poor or have been made poor by exploitation are abandoned and negative feedback behavior arises to balance the positive feedback.

There are total M nodes, and on each node, there are n_i VMs running. There are l_i number of maximum possible jobs running on each VM. For each job, function ABC-scheduler will run for MCN number of times, where MCN is the maximum number of cycles. This will result in the the system complexity of $O(M \times MCN)$ and process complexity for each process as $MO(n \times l)$.

This section delineated the detailed design, algorithm and the flowchart of the proposed ABC-based ERU. The next section elaborates the performance analysis of the proposed ERU model by using a CloudSim simulator followed by a detailed discussion on the efficacy of the proposed model over the existing FFD & ACO approaches.

3.4 Experimental Results of ERU Model

In this section, the performance evaluation of the proposed ABC-based ERU model has been presented. Thereby, the performance of ERU has been verified in a simulation-based experiment by using an existing cloud computing simulation framework, CloudSim toolkit [199] [200]. The CloudSim toolkit is a completely customizable tool used for seamless modeling, simulation, and experimentation of emerging cloud computing infrastructures and application services. It allows the researchers and the industry-based developers to focus on the specific system design issues that they want to investigate, without getting concerned about the low level details related to the cloud-based infrastructures and services. The CloudSim toolkit has been used for the evaluation due to the following reasons [199] [200] :

- It allows the modeling and the simulation of the large scale cloud computing data centers, the virtualized server hosts, with the customizable policies for provisioning the host resources to the virtual machines.
- It provides the support for the modeling and the simulation of the energy-aware computational resources.

- It supports for the modeling and the simulation of the data center network topologies and the message-passing applications
- It has a support for the dynamic insertion of the simulation elements, stop and resume of simulation
- It gives support for the user-defined policies for the allocation of the hosts to the virtual machines and the policies for the allocation of the host resources to the virtual machines

3.4.1 Performance Evaluation

Table 3.3 shows the specification details of implementation of the whole cluster which has been composed of homogeneous nodes, whereas Table 3.4 gives the description of a node & Table 3.5 gives the description of a VM. Up to 600 VMs have been simulated. The average power drawn by the system during simulation was 198 W, that has been measured with the help of Joulemeter [201], which is a software tool to measure the power consumption of the system. For estimating the energy consumption, a time period 't' has been defined and set to one day. Therefore, the energy consumption values represent the power drawn by the system over a time period of 24 hours. For optimized results and energy savings, an assumption has been made to turn-off the idle nodes after the workload consolidation. Hence, their idle powers have not been included in calculating the total energy consumption. In particular, the energy consumption has been estimated according to Equation 3.8.

Table 3.3: Simulation Parameters

Parameter	Value	Comment
No. of VMs	100-600	Supporting Cloud Environment
No. of Nodes	25-175	Nodes/Servers running VMs
Bandwidth	10 Gbps	Maximum allowed data rate
Average Energy Consumed	456 kWh	Average energy consumed during completion of jobs
Average Computational Energy	52 Wh	Average energy spent for computing the placements of jobs
Average Power drawn	198 W	Power rate of nodes
No. of Employed Bees	25-175	Equals number of food sources (M)
Maximum Cycle Number	1000	Maximum Cycle Number of the search process
Limit	$M \times D$	Number of predetermined cycles for improvement phase where M is the number of food sources and D is the number parameters to be optimized

Table 3.4: Node Description

Parameter	Value	Comment
Disk Storage	1 TB	Physical/Secondary memory available
RAM	50 GB	Primary memory available
No. of cores	24	Number of cores in the node
CPU Capacity	10000 MIPS	Number of CPU cycles

Table 3.5: VM Description

Parameter	Value	Comment
Disk Storage	200 GB	Physical/Secondary memory assigned to a VM
RAM	4 GB	Primary memory assigned to a VM
No. of cores	2	Number of cores assigned to a VM
CPU Capacity	1000-5000 MIPS	Number of CPU cycles assigned to a VM
Bandwidth	1 Gbps	Maximum allowed data rate

The simulation has been carried out for 100 to 600 VMs and is repeated 30 times. The Execution Time (ET), Energy Consumed (EC), Computational Energy (CE), percentage of nodes and energy saved have been calculated for the proposed ABC-based ERU. The calculated results have been compared to FFD and ACO-based techniques to depict the percentages of saved nodes and energy using ERU over the other two techniques. The FFD approach sorts the jobs in the decreasing order before processing them and the ACO approach processes the jobs using its probabilistic nature. The standard evaluation values for the proposed model have been considered similar to as taken in the comparison of FFD and ACO by Feller et al. [130], for the true analysis on the same conditions.

Figure 3.3 illustrates the comparison of Total ET of all the three approaches. In this Figure, bars represent the number of nodes used by each technique whereas lines represent the execution time of each technique. The Total ET, which is the total time to get the final output is measured in minutes (mins). The Figure shows that the Total ET of ERU is greater than FFD and lesser than ACO. FFD has outperformed the other two, because jobs have been assigned to VMs as soon as they entered the system, whereas ACO & ERU take time in finding the appropriate best solution. Being based on ABC, the Total ET of ERU is better than ACO as the convergence rate of ABC is high. This is because the Convergence Rate (C_{rate}) is inversely related to the Execution Time (ET), i.e. $ET \propto (1/C_{rate})$. In other words, the higher the rate of convergence, the lower the execution time. Moreover, ABC based ERU often uses higher fitness

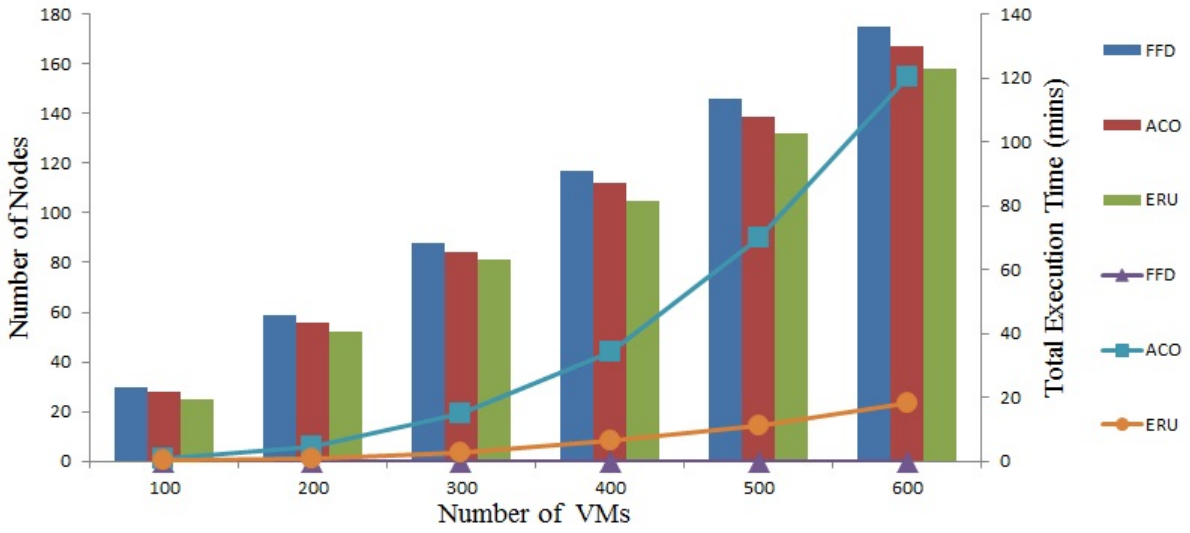


Figure 3.3: Total ET: ERU vs FFD vs ACO

value solutions than those with smaller/lower fitness values to produce trial solutions leading to a shorter time of search.

Figure 3.4 shows the comparison of CE for ERU, ACO and FFD. Here again, bars represent the number of nodes used whereas lines represent the computational energy. CE is the energy spent for computing the placement. It is calculated as $(ET \times 198)$ and is measured in terms of Watt hours (Wh). This Figure presents that ERU consumes less energy in computation than ACO as ERU's execution time is less than the execution time of ACO. ERU consumes more energy in computation than FFD as its execution time is more than FFD. CE has been incorporated into the total energy consumption and has been calculated not more than 60 Wh. Therefore, it has not influenced the total energy results of the algorithm which have been in the region of kWh.

Figure 3.5 depicts the comparison of EC of ERU with that of the other two techniques, i.e. EC of FFD and ACO. The bars in the Figure represent the number of used nodes whereas lines represent the energy consumed. The EC is measured in terms of kilo Watt hours (kWh). The figure illustrates the energy efficiency of ERU as compared to FFD and ACO based techniques. The EC of the ERU is lowest as compared to ACO and FFD. The proposed ERU has outperformed both the techniques as it uses less number of nodes than the other two. Also, ERU converges faster than ACO but slower than FFD as ERU's ET is less than ACO but more than FFD. This is because $C_{rate} \propto (1/ET)$ i.e. the decrease in execution time will result in an increase in the convergence rate. Therefore, due to the usage of less number of nodes and high conver-

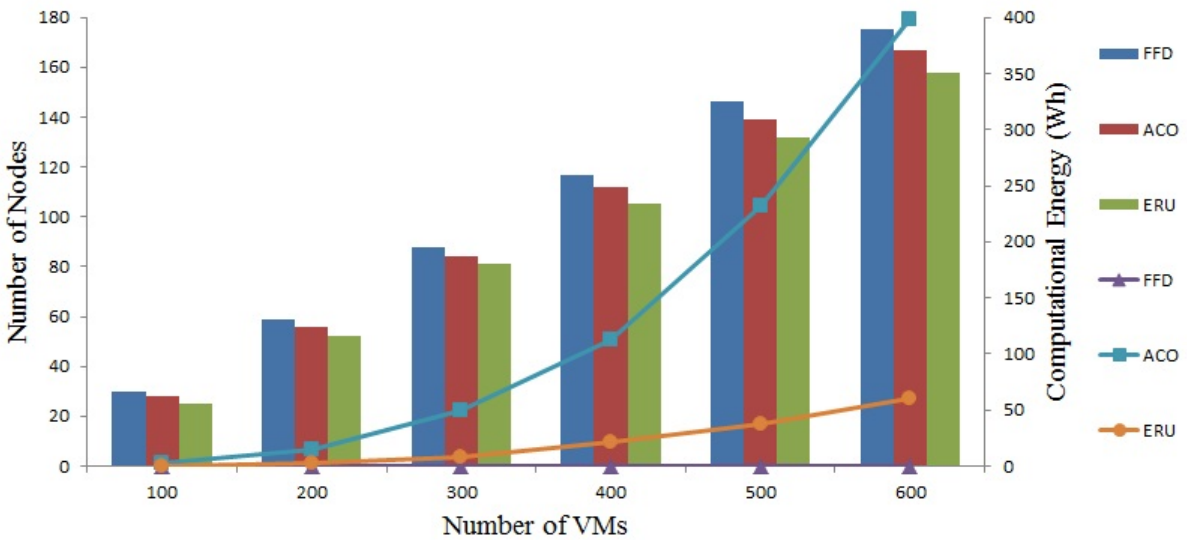


Figure 3.4: CE : ERU vs FFD vs ACO

gence rate, EC of ERU is lowest for all experimental iterations. But for FFD, besides having the least ET, its EC is more, as FFD does not utilize the resources properly leading to the usage of more number of nodes than the other two approaches.

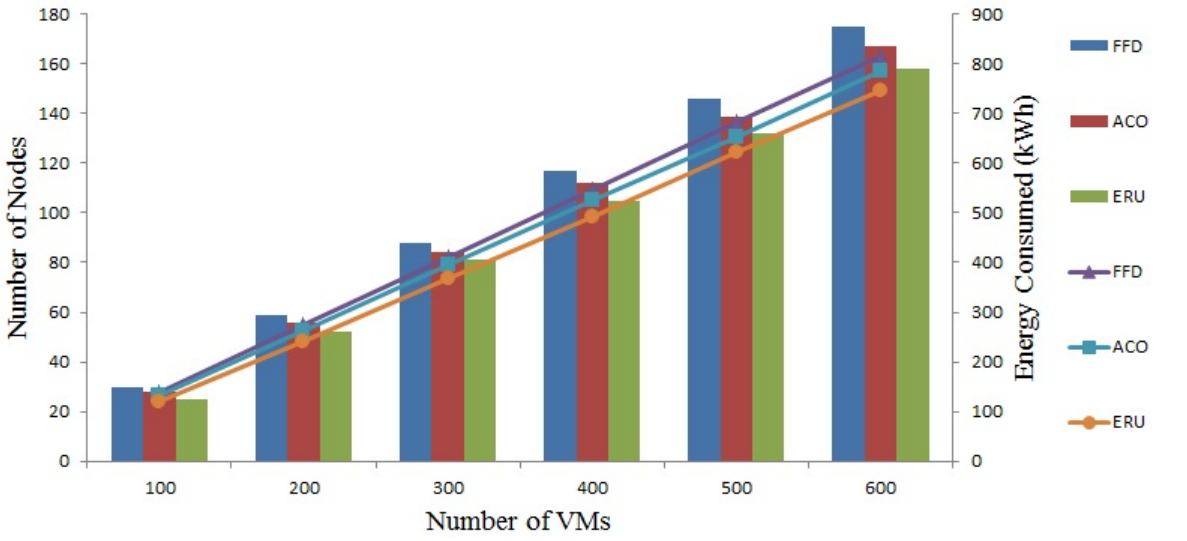


Figure 3.5: EC : ERU vs FFD vs ACO

3.4.2 Discussion

The efficiency of the proposed ERU has been highlighted in Figure 3.6, which clearly states and justifies the credibility of the proposed ERU model. The results demonstrate that an average of 8.68% of nodes and 8.66% of energy has been conserved using ERU

over FFD and ACO-based techniques. The ABC-based ERU has further been compared individually to FFD and ACO. Though FFD has outperformed ERU in terms of both execution time and computational energy, but nodes and energy saved are far better in ERU than FFD. On an average 11% of nodes and 10.7% of energy have been conserved using ERU over FFD based technique. The ERU has outperformed ACO in every way and the nodes and energy saved using ERU over ACO technique are 6.35% and 6.63% respectively.

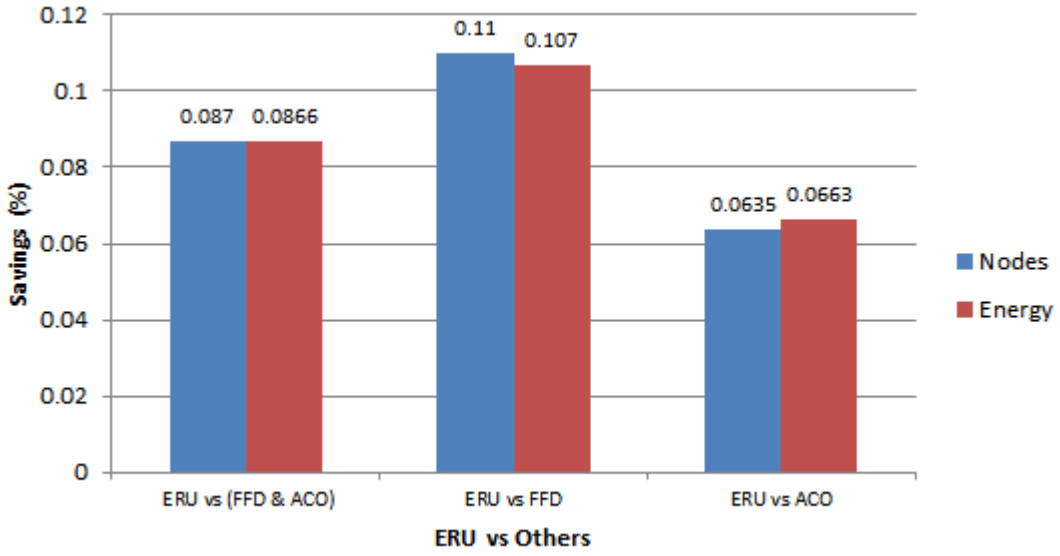


Figure 3.6: Improvement Results of ERU over Other Techniques

Thus, the ERU Model can assist the organizations in enhancing the customer satisfaction and contributes directly to the green environment by reducing the energy consumption and hence the carbon emission.

This section delineated the performance analysis of the proposed ERU model by using a CloudSim simulator followed by a detailed discussion on the efficacy of the proposed model over the existing FFD & ACO approaches. This section elaborates the validation of the proposed ERU with an existing approach.

3.5 Validation of ERU with Existing Technique

This section presents the comparative view of the proposed ERU and an existing ESWCT (Energy-aware Scheduling using Workload-aware consolidation Technique) [202] using the same simulation environment as given in the above section. ESWCT algorithm deals with the admission of new applications for VM provisioning

and places the VM on the node. It tries to execute all VMs with minimum amount of nodes, and then power off unused nodes to reduce power consumption [202].

The number of hosts and the energy consumption values used by both the techniques have been compared as shown in Figure 3.7 & Figure 3.8. Figure 3.7 presents the comparative view of the proposed ERU technique with the ESWCT approach on the basis of the required number of hosts over the number of VMs as an independent axis. The optimal allocation of resources is important to enhance the resource utility levels thereby enhancing energy efficiency. The contention of the resources should also be considered to avoid performance degradation and to enhance energy efficiency.

It is clear from the analysis of the obtained results that ERU uses lesser number of nodes than ESWCT. This is due to the fact that ERU is based on the ABC optimization technique that efficiently manages the cloud resources and optimally allocates VMs on the minimum number of active nodes with reduced execution time resulting in reducing the number of active nodes and increasing their utility levels. A global optimization is also achieved at a high convergence rate.

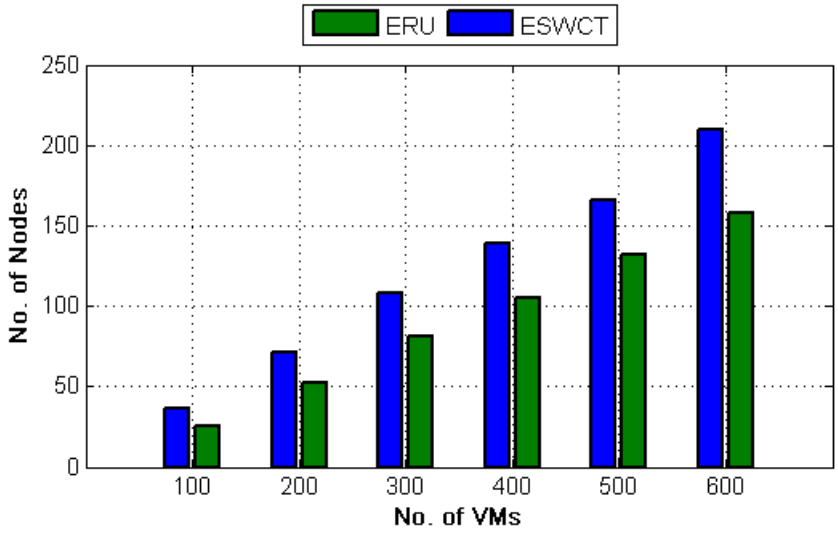


Figure 3.7: VMs vs Nodes

Figure 3.8 depicts the comparison of energy consumed by ERU with that of ESWCT. The energy efficiency of ERU as compared to ESWCT is illustrated by this figure. The energy consumption of ERU is less than that of ESWCT as ERU uses less number of nodes than ESWCT thereby increasing their utilization. As less utilized or idle nodes consume power, therefore minimizing the number of active nodes and enhancing their utilization avert energy wastage and enhance energy efficiency.

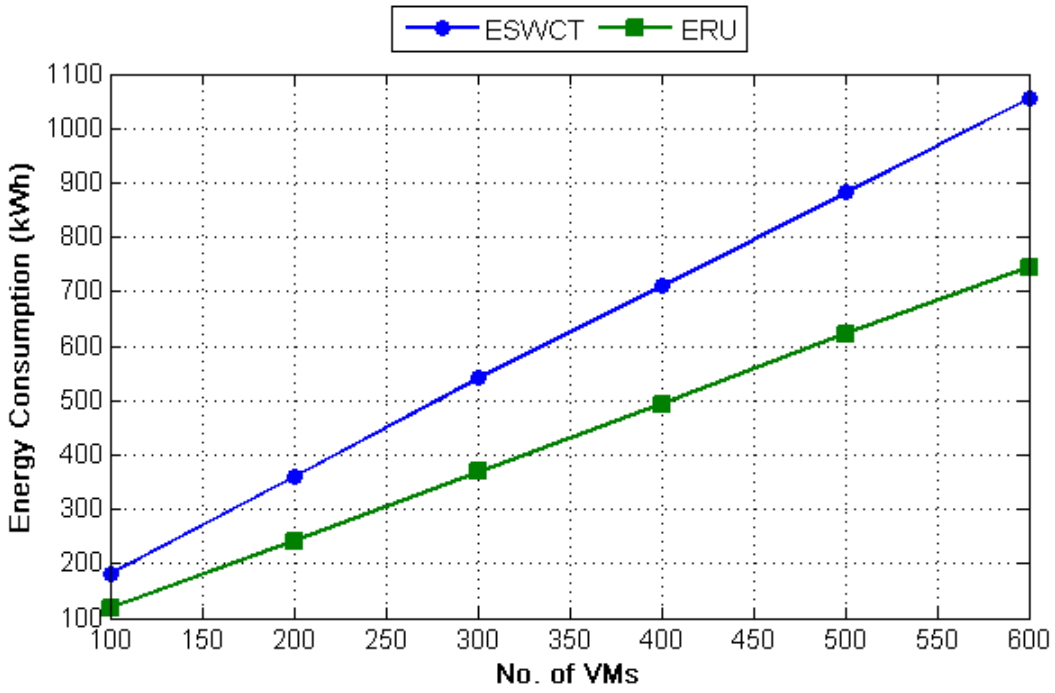


Figure 3.8: VMs vs Energy Consumption

3.5.1 Discussion

The effectiveness of the proposed ERU has been highlighted in Figure 3.9. The results clearly indicate that an average of 25.34% of active nodes and 31.33% of energy have been saved using the proposed ERU over ESWCT approach.

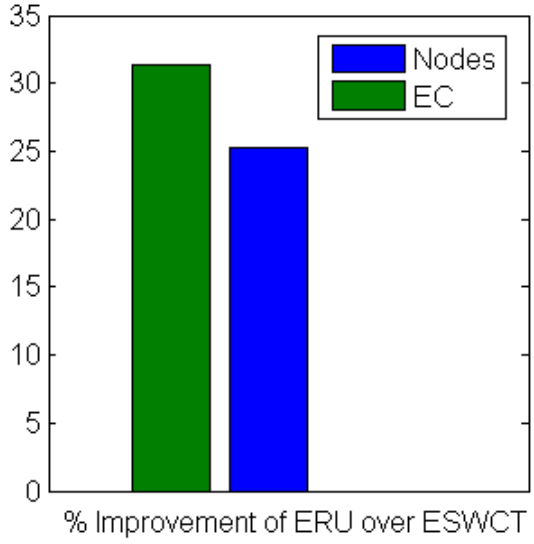


Figure 3.9: Improvement Graph of ERU over ESWCT

3.6 Conclusion

This chapter discussed the proposed ERU model with its detailed requirements. It also presented ABC optimization technique and its mode of operation in the model. The chapter also elaborated the performance analysis of the proposed ERU model using CloudSim simulator. The results demonstrated that an average of 8.68% of nodes and 8.66% of energy has been conserved using ERU over FFD and ACO-based techniques. This chapter further presented the validation of ERU with an existing ESWCT approach. The results indicated that an average of 25.34% of active nodes and 31.33% of energy have been saved using the proposed ERU over ESWCT approach.

The next chapter presents the design of the proposed Firefly optimization based Energy-aware VM migration (FFO-EVMM) technique to lower the consumed energy in cloud data centers.

Chapter 4

Proposed Energy-aware VM Migration Technique

The previous chapter discussed in detail the model and design of ABC optimization based ERU model to enhance the utilization of cloud resources to achieve energy-efficiency in cloud data centers. Further, during execution i.e. at run-time, if energy consumption increases, VM migration is required, which is an extended facility provided by virtualization to support the efforts for realizing energy efficient computing.

VM migration helps to manage workloads on individual nodes by migrating running VMs without any perceptible downtime, from heavily-loaded nodes to lowly-loaded nodes, thereby obtaining energy optimization by minimizing energy consumption. This chapter presents the proposed VM Migration Technique and puts forward a FireFly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) technique for cloud environment to reduce consumed energy in cloud data centers at run-time. FFO-EVMM technique intends to maximize energy-efficiency through optimum migration of VMs, thereby improving resource utilization levels.

This chapter initially presents Firefly optimization technique being the basis of the proposed VM migration technique. Further, it provides the design of FFO-EVMM, its description, flowchart and performance analysis over FFD & ACO in the CloudSim toolkit. Finally, it validates FFO-EVMM with an existing approach.

4.1 Firefly Optimization (FFO) Algorithm

Firefly Optimization (FFO) algorithm has been designed by Xin-She Yang in late 2007 and 2008 at Cambridge University [165] [182] [183]. It is centred on the social behavior of fireflies and the phenomenon of bioluminescent communication. The flashing features of fireflies are due to the presence of photogenic organs close to their body surface. These bioluminescent signals tend to attract the attention of other fireflies. The intensity of the light signal emitted from each firefly determines its ability to explore an efficient new solution. The Firefly Optimization (FFO) algorithm uses the subsequent three idealized procedures: (1) One firefly is attracted to the other fireflies irrespective of their sex as all the fireflies are unisex, (2) The attractiveness is proportionate to the brightness, thus they both decrease as their distance increases and for any two flashing fireflies, the less brighter one travels near the brighter one. If no firefly is brighter than a specific firefly, it moves arbitrarily and (3) The brightness of a firefly is regulated by the landscape of the objective function to be optimized.

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, therefore, the variation of attractiveness β with the distance r can be defined as [165] [182] [183] :

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4.1}$$

where β_0 is the attractiveness at $r = 0$. The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by [165] [182] [183] :

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha_t \epsilon_i^t \tag{4.2}$$

where α_t is the randomization parameter that controls the randomness and ϵ_i^t is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. The second term is due to the attraction and the third term is due to randomization. If $\beta_0 = 0$, it becomes a simple random walk. On the other hand, if $\gamma = 0$, it reduces to a variant of particle swarm optimization (PSO) [165] [161].

This section outlined the FFO technique being the basis of the proposed VM migration technique. The next section elaborates the design of the proposed FireFly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) technique along with its detailed description and the pseudoflow of its algorithm.

4.2 Proposed FireFly Optimization based Energy-aware Virtual Machine Migration (FFO-EVMM) Technique

This section proposes an energy-aware virtual machine migration technique for cloud computing that is based on the flashing behavior of fireflies. This technique tries to migrate the most loaded VM from an active node which satisfies a minimum criterion for energy consumption, to another active node that consumes the least energy. It consists of the four main parts, A) Selection of source node, B) Selection of VMs, C) Selection of destination node and D) Distance updated values. The description of all the above parts is as follows :

A. Selection of source node: The source node is the active node from where, VMs have to be migrated. The active node, which is at the least distance(defined in step 4.) from the destination node, is selected as the source node. For this, the following steps are done.

Step 1. Energy Consumption (EC) of each active node is calculated using equation 4.3 and then, all the values are stored in a list.

1. Compute EC for each active node using the following equation

$$EC_i = (\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times t \tag{4.3}$$

   where n_i is the number of VMs running on the i^{th} node and l_i is the number of jobs assigned to n_i VMs. rpu_{ijk} and rmu_{ijk} are the processor and the memory utilizations of k jobs running in j VMs on the i^{th} node respectively.

2. Store these values in a list, EC.

Step 2. Time-based optimization : After computing the EC of each active node, the next step is to optimize the proposed technique for the performance in terms of minimizing the Execution Time (ET). The ET is calculated for each active node using equation 4.4 and the values are again stored in a list.

1. Compute Execution Time (ET) for each active node using the following

equation

$$ET_i = \sum_{j=1}^{n_i} \sum_{k=1}^{l_i} ET_{ijk} \tag{4.4}$$

where ET_{ijk} is the execution time of k jobs running in j VMs on the i^{th} node.

2. Store these values in a list, ET.

Step 3. Attraction Index (AI) : The attraction property of the fireflies has been modelled by computing an AI value. The AI value is calculated using Indexed based searching and a sorted AI list is prepared according to EC values. The active node with the least EC is obtained as the first element of the list.

1. Compute $AI_i(EC_i, ET_i)$ for each active node,

   where AI_i is the Attraction Index, EC_i is the energy consumption and ET_i is the Execution Time for the i^{th} node respectively.

2. Store these values in a list, AI and sort this list in an ascending order according to the EC values.

Step 4. The node to be selected as a potential source for VM migration, must satisfy a minimum criterion for the energy consumption and this is controlled by a distance value which is computed by using equation 4.5. When finding a solution, this distance has to be the least in order to keep the energy consumption to the minimum. Thus, the node is selected, which has the EC value nearest to the computed distance value, to be the source node from where the VMs will be migrated.

1. Compute Distance, using the following equation :

$$Distance = Avg(AI_{mid}, AI_{max}) \tag{4.5}$$

   where AI_{mid} and AI_{max} are the middle and the maximum values from the AI list.

2. Select the node with EC value closest to the above computed $Distance$ value.

B. Selection of VMs : The VMs to be migrated are determined.

Step 5. To select the VMs to be migrated from the source node, calculate the load of each VM on the source node according to equation 4.6. Then, these values are stored in a list and the list is arranged from higher load value to the lower load value. The VM with the highest load value is selected to be moved to the destination node.

1. Compute Load for each VM of the above selected node at time instance Δt, using the following equation :

$$Load_{ij} = \frac{\sum_{j=1}^{n_i} job_{ij}}{(\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times \Delta t} \tag{4.6}$$

   where job_{ij} represents the total number of jobs running in the j^{th} VM on the i^{th} node and $(\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk})$ is the total consumed power of the i^{th} node in Δt units of time.

2. Store these values in a list, *Load* and sort this list in the descending order.

3. Move the first element of the obtained list, *Load* to the first element of the *AI* list, i.e. move the most loaded VM towards the most brightest node.

C. Selection of destination node: The destination node for the VMs is discovered.

Step 6. Discovering the most brightest node : The property of fireflies to move towards the brighter nodes requires to identify the brighter node. The node is said to be the brightest if its Energy Consumption (EC) is minimum. The active node with the least EC value, is obtained as the first element of the AI list and is thus the primary contender for migrating the overloaded VMs to it.

D. Distance Updated Values : The distance values are updated.

Step 7. The updation involved after each iteration is the updated value for the dis-

tance, which is given by Equation 4.7 as follows :

$$(Distance)^{t+1} = (Distance)^{t} + \frac{\sum_{j=1}^{n_i} job_{ij}}{(\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times \Delta t} + \epsilon \quad (4.7)$$

where $(Distance)^{t}$ & $(Distance)^{t+1}$ are the distance values at time t & $t+1$, second term is due to the *Load* and ϵ is the gaussian distribution error. The corresponding pseudocode is given in Algorithm 2 and the flowchart for FFO-EVMM technique has been presented in Figure 4.1.

Algorithm 2: FFO-EVMM Technique Algorithm

Data: Scheduling Information consisting of set of jobs, set of nodes, resource utilization & energy information of nodes(Chapter 3)
Result: Finding the best VM-node pair
begin
- *Source_Node*()
- **for** *(each Node)* **do**
  - Compute Energy Consumption using Equation 4.3
  - $EC[] \leftarrow$ Energy Consumption value
  - Calculate Execution Time using Equation 4.4
  - $ET[] \leftarrow$ Execution Time value
- **for** *(each Node)* **do**
  - Compute Attraction Index
  - $AI[] \leftarrow$ Attraction Index value
- Sort $AI[]$ in an ascending order according to Energy Consumption values
- Compute Distance using Equation 4.5
- Find the node with EC value nearest to the calculated Distance value from the sorted $AI[]$ list
- *Culprit_VM*()
- **for** *(each VM on the Source Node)* **do**
  - Compute Load using Equation 4.6
  - $Load[] \leftarrow$ Load value
- Sort $Load[]$ in a descending order
- *Destination_Node*()
- Get the first element of sorted $AI[]$ list as the destination node
- Move the first element of $Load[]$ list to the first element of $AI[]$ list
- Update the Distance value using Equation 4.7

An example has been provided to have a better understanding of the algorithm. Suppose that there are 5 active nodes (N_1....N_5) and overall 12 VMs (VM_1....VM_{12}) are running on them. Let the VMs VM_1 and VM_3 be running on the node N_1 which is represented as $N_1 = (VM_1, VM_3)$, and same is for all the other four nodes, i.e. $N_2 = (VM_2, VM_4, VM_6)$, $N_3 = (VM_5, VM_7)$, $N_4 = (VM_8, VM_{10}, VM_{11})$ and $N_5 = (VM_9, VM_{12})$. EC values of these active nodes are calculated using Equation 4.3 and ET values of these active nodes are calculated using Equation 4.4 as given in Table 4.1. These values are then used to find AI values.

Table 4.1: Example

Nodes	EC Values (Wsec)	ET Values (sec)	AI Values
N_1	1308.89	0.36	$AI_1 = (1308.89, 0.36)$
N_2	2507.24	0.38	$AI_2 = (2507.24, 0.38)$
N_3	1764.06	0.42	$AI_3 = (1764.06, 0.42)$
N_4	2872.72	0.49	$AI_4 = (2872.72, 0.49)$
N_5	1922.52	0.53	$AI_5 = (1922.52, 0.53)$

The AI values are stored in a list which is sorted in an ascending order of EC, to obtain another list as ($AI_1 = (1308.89, 0.36)$, $AI_3 = (1764.06, 0.42)$, $AI_5 = (1922.52, 0.53)$, $AI_2 = (2507.24, 0.38)$ and $AI_4 = (2872.72, 0.49)$). Now, the distance value is computed as the average of $AI_5 = (1922.52, 0.53)$ and $AI_4 = (2872.72, 0.49)$ using Equation 4.5, which is equal to $(1922.52 + 2872.72)/2 = 2397.62$ Wsec.

The next step is to select that node from the sorted AI list whose EC value is nearest to 2397.62 Wsec, which is the node having value 2507.24, that is node N_2 which becomes the source node from where the VMs are migrated. Now the load of each VM on the node N_2 is calculated using Equation 4.6 as $VM_2 = 835.75$ Wsec, $VM_4 = 1253.62$ Wsec and $VM_6 = 417.87$ Wsec. These values are stored in a list and that list is sorted in descending order and a list is obtained as (VM_4, VM_2, VM_6). Therefore, VM_4 becomes the VM to be migrated. Next, the destination node is chosen where this VM is moved. As the destination node is the node with the least EC value, it is obtained from AI list where the nodes are arranged in an ascending order of EC. So, the first element of the sorted AI list, that is node N_1 is the destination node where VM_4 is migrated and the distance value is updated using Equation 4.7 for the next iteration.

Whenever a running VM needs to be migrated, the entire state of the VM (embracing the virtual CPUs, the drivers' configuration, the memory of VM and the storage) is relocated [203]. For the efficient VM migration and the centralized availability, the

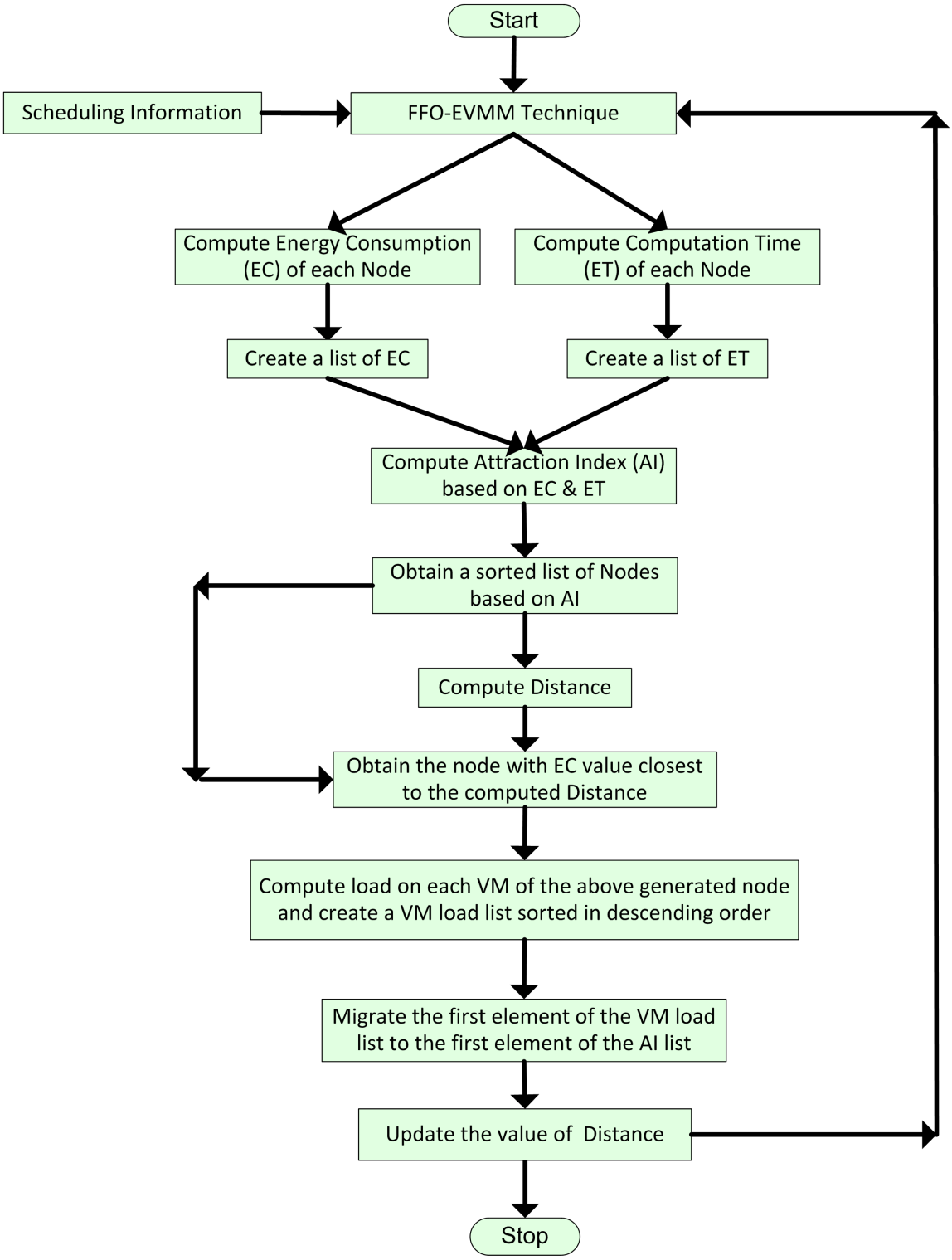


Figure 4.1: Flowchart for FFO-EVMM Technique

images and data of all the VMs have been stored on a common storage called the NAS. NAS is reachable to all the nodes and serves as a storage for the VMs where non-redundant data is stored [204]. Whenever a VM is migrated, only in-memory states and CPU registers of that VM need to be migrated from one node to the other as its image and storage contents are accessed from NAS [203] [204]. This saves the storage space, improves the search rate and the look up time, gears up the flow rate of the data during a VM migration. This further helps in diminishing the data transfers and hence the time taken to migrate a VM, thereby dropping the consumed energy and the associated energy expenditures incurred while migrating VMs from one node to another. The faster execution of the tasks results in the minimization of the overall system execution time and hence improves the performance. Wholely, FFO-EVMM with the use of NAS, tends to deal with the overheads involved in VM migrations.

This section outlined the detailed design, algorithm and the flowchart of the proposed FFO-EVMM technique. The next section delineates the performance analysis of the proposed FFO-EVMM technique by using CloudSim simulator followed by a detailed discussion on the effectiveness of the proposed technique over the existing FFD & ACO approaches.

4.3 Experimental Results of FFO-EVMM Technique

This section evaluates the proposed FFO-EVMM algorithm and compares it with the ACO-based and FFD-based algorithms, using the CloudSim toolkit [200]. The details of the CloudSim toolkit has been given in Chapter 3.

4.3.1 Performance Evaluation

Table 4.2, Table 4.3 & Table 4.4 give the specification details of the simulation parameters, a node and a VM respectively. Up to 200 VMs and 200 nodes (fireflies) have

Table 4.2: Simulation Parameters

Parameter	Value	Comment
No. of VMs	20-200	Backing Cloud Environment
No. of Nodes	10-200	Nodes running VMs
Bandwidth	2.5 Gbps	Maximum allowed data rate

been simulated and the simulation is repeated for multiple runs.The simulations have been carried out by using the same evaluation parameters as defined by Beloglazov et al. [146].

Table 4.3: Node Description

Parameter	Value	Comment
Node_Types	2	Types of Node used
Node_PES	2	Dual-core Node
Node_MIPS	1860-2660	MIPS allocated to each Node
Node_RAM	4GB	Primary Memory allocated to each Node
Node_Storage	1 TB	Secondary storage allocated to each Node

Table 4.4: VM Description

Parameter	Value	Comment
VM_Types	4	Types of VMs used
VM_PES	1	Single-core VMs
VM_MIPS	500-2500	MIPS allocated to each VM
VM_RAM	.5GB-4GB	Primary Memory allocated to each VM
VM_SIZE	2.5GB	Secondary Memory allocated to each VM

The Consumed Energy (EC), the number of saved nodes and the number of reduced migrations have been calculated through the proposed FFO-EVMM technique. The energy consumption has also been computed using different node utilization thresholds. The obtained results have been compared to the ACO-based and the FFD-based techniques and are shown in Figure 4.2 to Figure 4.10.

Figure 4.2 presents the comparative view of the three techniques, that is, FFD, ACO and FFO-EVMM on the basis of the required number of active nodes over the number of VMs as independent axis. It is important to keep a track of the number of nodes operating in the system in order to prevent situations where the probability of most of the nodes sitting idle and consuming unnecessary power is high, which will violate the minimum energy requirement criterion. Upon identification of idle nodes, they are set to sleep mode.

Based on the analysis of the obtained results, it is evident that FFO-EVMM technique runs lesser number of active nodes in comparison to the other two techniques. This is because FFO-EVMM runs firefly optimization algorithm that chooses accurate nodes for VM migration with reduced discovery time resulting in an optimal utilization of the host nodes. It attains global optimization with faster convergence speed. The

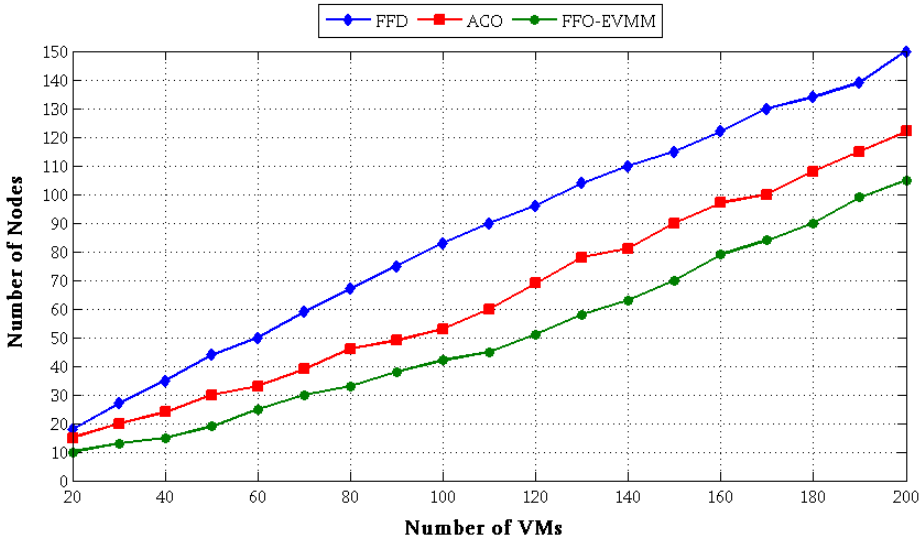


Figure 4.2: VMs vs Nodes

improvement in the resource utility levels minimizes the number of VM migrations thereby averting energy wastage.

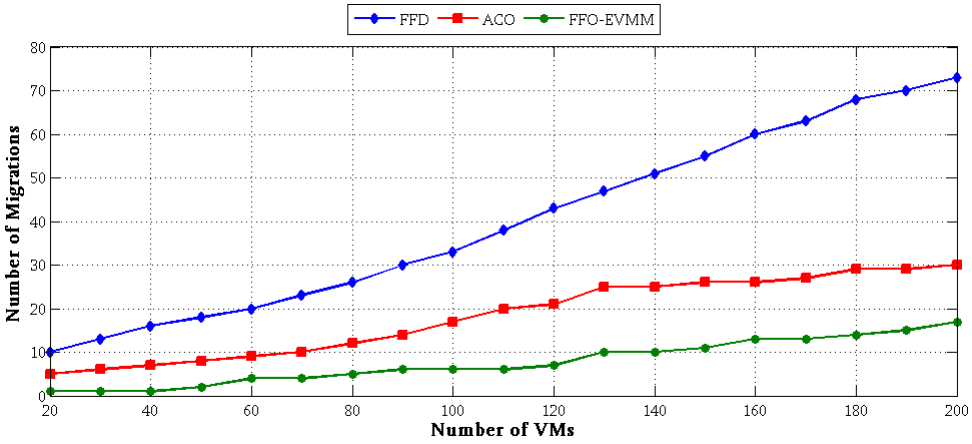


Figure 4.3: VMs vs Number of Migrations

The same is shown through Figure 4.3. The graph in Figure 4.3 depicts the number of VM migrations performed by the three techniques. As observed in Figure 4.2 and Figure 4.3, FFO-EVMM technique uses lesser number of nodes and performs lesser number of VM migrations in contrast to FFD and ACO. The capability of FFO-EVMM to pro-actively discover the best node for VM migration without compromising with the energy consumption affects the future migration decisions and the number of VM migrations required. The overall energy consumption in the system is optimal upon the arrival of the new workload, as all the previous workload allocations to the VMs running

on the system nodes have been done considering the energy thresholds. Thereby, the need to incur more and more VM migrations is low while making further workload allocations as per the energy constraints.

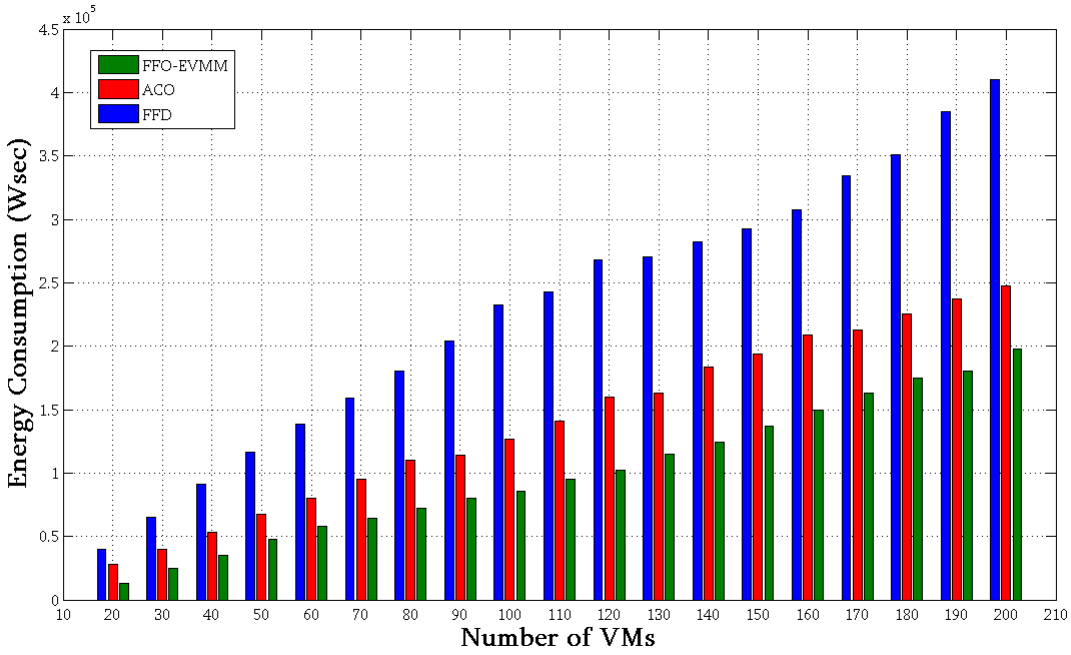


Figure 4.4: VMs vs Energy Consumption

With the lesser number of required nodes and VM migrations, FFO-EVMM poses lesser energy demand. The energy consumption done by FFO-EVMM is low as compared to FFD and ACO as observed from the graph given in Figure 4.4. The tendency to discover and reduce the number of active but idle nodes curtails the energy demand. It attains the efficient resource utility levels affecting the number of VM migrations. The reduction in the number of VM migrations cuts down the amount of the energy consumed, that would have otherwise been wasted when the VMs are being migrated. Consequently, the required operational energy and the energy consumption level drops down.

Figure 4.5 depicts the average energy consumption at different threshold values according to the host utilization levels for all the three techniques. It is visible from the graph that as the utilization of the nodes increases from 20% to 80%, the threshold values also vary accordingly. Upon simulation, it has been concluded that at different values of thresholds, in FFD and ACO based techniques, there is a fluctuation in the energy consumption done by FFD and ACO. The energy consumed by FFD and ACO in comparison to the proposed FFO-EVMM technique follows an irregular behaviour,

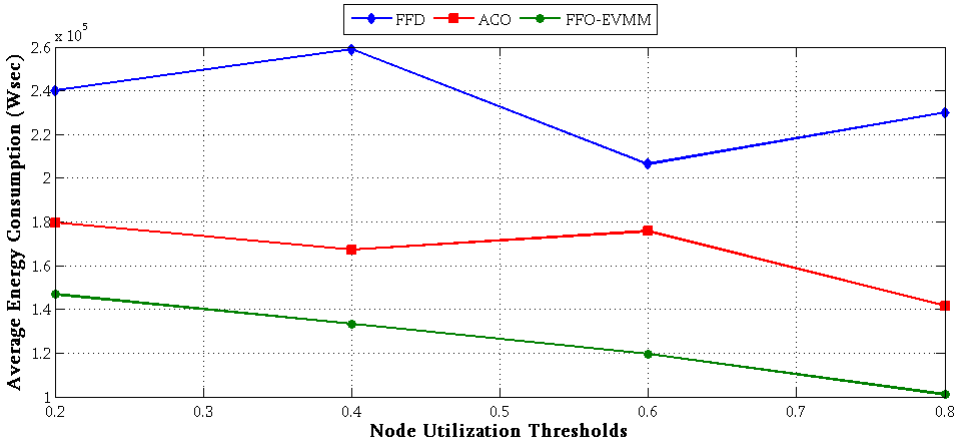


Figure 4.5: Thresholds vs Avg Energy Consumption

that is, the energy consumption keeps on increasing or decreasing for different threshold values. Relatively, the energy consumption in FFO-EVMM technique decreases consistently with varying thresholds. The declining trend in the energy consumption of FFO-EVMM is due to the improvement attained in the host utility levels because of energy-aware VM migration decisions.

4.3.2 Discussion

Figure 4.6 to Figure 4.9 vindicate the reliability and competence of the proposed FFO-EVMM technique. Figure 4.6 indicates the overall enhancement of the proposed tech-

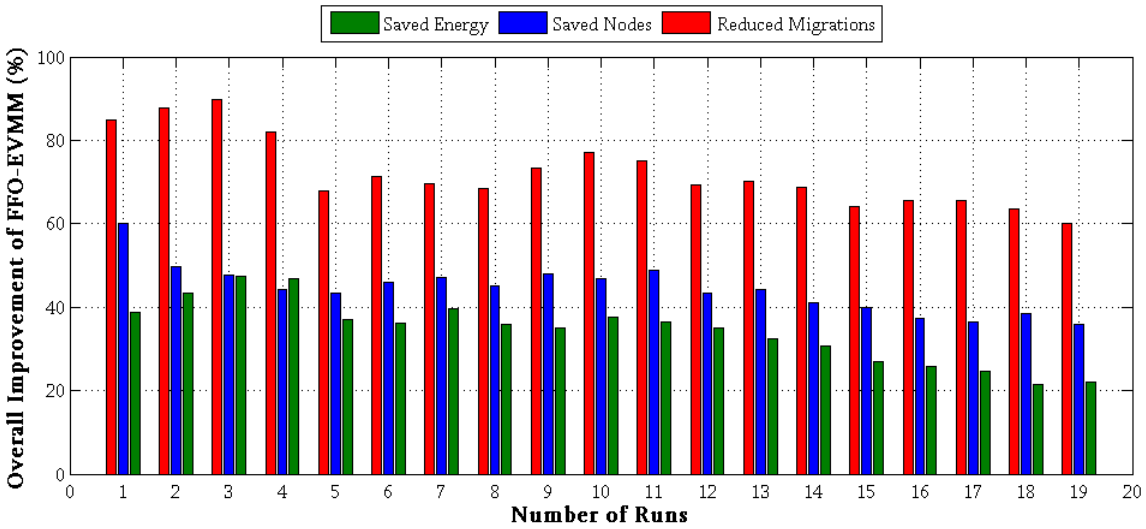


Figure 4.6: Overall improvement of FFO-EVMM

nique over the other two. The outcomes determine that an average of 72.34% of migrations have been reduced and 34.36% of nodes have been saved with FFO-EVMM. Due to the less number of migrations and nodes, an average of 44.39% of energy has been saved using FFO-EVMM over ACO-based and FFD-based techniques. Further, the comparison of the proposed FFO-EVMM technique has been done to FFD-based and ACO-based techniques separately, to get an insight into its efficacy over each technique.

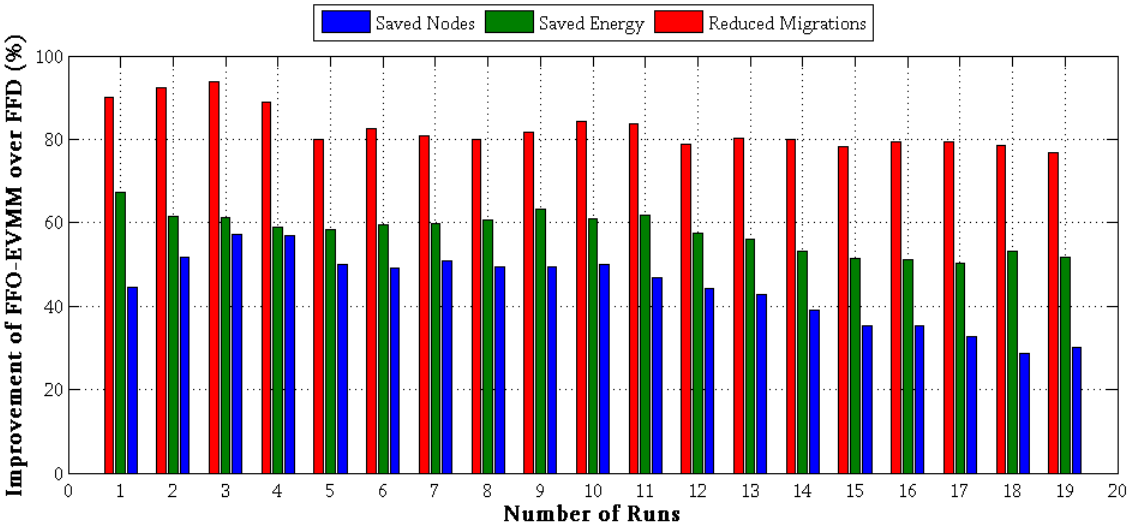


Figure 4.7: Improvement Graph of FFO-EVMM over FFD

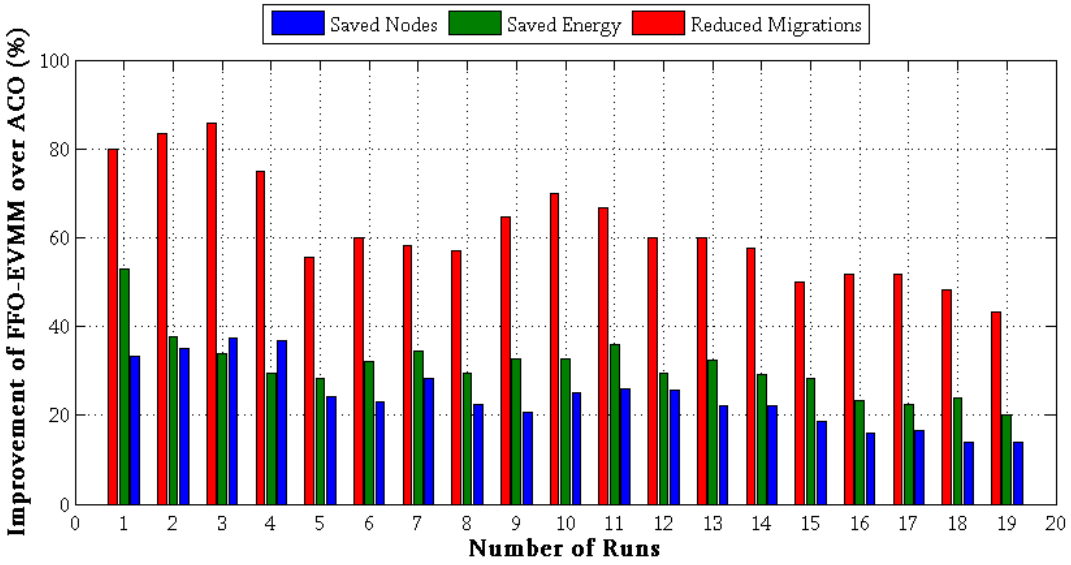


Figure 4.8: Improvement Graph of FFO-EVMM over ACO

When, FFO-EVMM technique is compared to FFD-based technique as shown in Figure 4.7, on an average, a reduction of 82.61% migrations and a saving of 44.43% nodes & 57.77% energy have been observed. Figure 4.8 denotes the comparison of FFO-EVMM over ACO-based technique indicating the percentage of saved nodes and energy as 24.29% & 30.99% respectively, whereas the percentage of the reduced migrations is 62.07%.

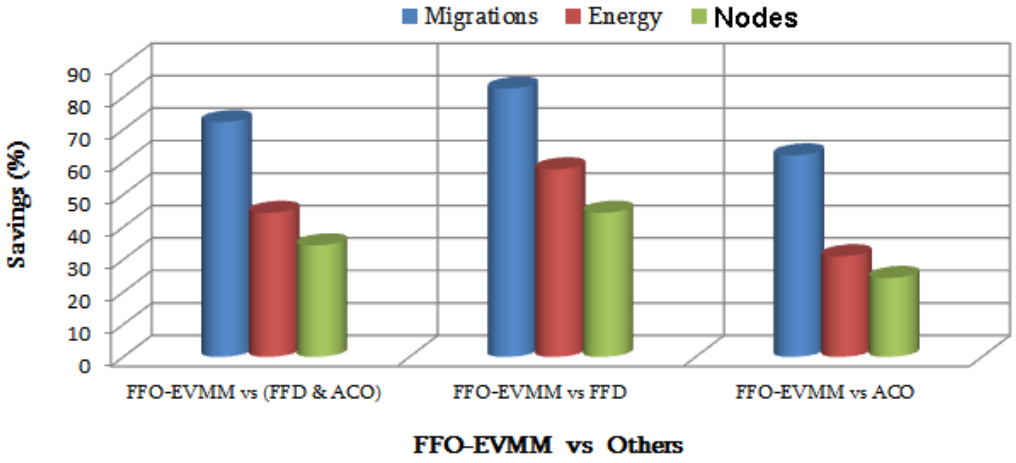


Figure 4.9: Percent Improvement of FFO-EVMM over Other Techniques

Figure 4.9 gives the overall and individual saving percentage of FFO-EVMM over the rest. The improvement in the energy consumption and the decrease in the number of nodes state the effectiveness of the proposed solution showing the optimization with reduced number of migrations. The performance evaluation of VM migration illustrates

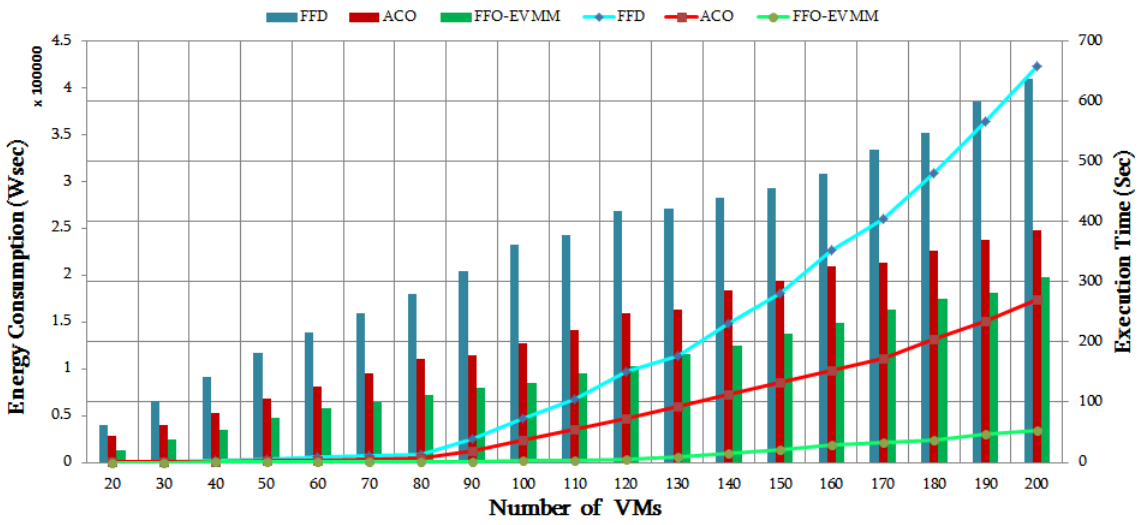


Figure 4.10: Performance Analysis

that the proposed schema can effectively be used for the larger cloud environment, thus making it highly scalable.

Figure 4.10 shows the run time analysis for all the three approaches. It can be interpreted from the graph that FFO-EVMM has outperformed the other two techniques. Being based on FFO, the execution time of the proposed technique is better than ACO and FFD as the convergence rate of FFO is very high. The trending curve shows the improvement in execution time of the overall applicability of the proposed algorithm. The lower execution time also decreases the overall complexity of the system.

This section depicted the performance analysis of the proposed FFO-EVMM technique by using a CloudSim simulator followed by a detailed discussion on the competence of the proposed technique over the existing FFD & ACO approaches. The next section drafts the details of the validation of FFO-EVMM over an existing technique.

4.4 Validation of FFO-EVMM with Existing Technique

This section presents the comparative view of the proposed FFO-EVMM and an existing MPSO (Modified Particle Swarm Optimization) based VM migration approach [205] using the same simulation environment as given in the above section. The MPSO based VM migration approach triggers a VM migration based on a method of double threshold with multi-resource utilization. It tries to minimize the consumed energy with the guarantee of Quality of Service [205].

The number of nodes, the number of VM migrations and the energy consumption values of both the techniques have been compared and presented as shown in Figure 4.11, Figure 4.12 & Figure 4.13 respectively.

The comparative view of FFO-EVMM with that of the existing MPSO based technique, on the basis of the required number of active nodes, has been presented in Figure 4.11. A detailed investigation of the obtained results indicate that the number of active nodes increases with the increase in the number of VMs using both FFO-EVMM and MPSO techniques. However the number of active nodes is less in MPSO in comparison to the FFO-EVMM technique. This is due to the reason that MPSO technique sorts all VMs in decreasing order of their current CPU utilizations and allocates each VM to an active node that contributes the least increase of power consumption due to this allocation. Whereas this sorting is not done in case of the

FFO-EVMM approach, however VMs are allocated optimally considering the resource contention, thereby avoiding unnecessary migrations, which is not a case in MPSO technique.

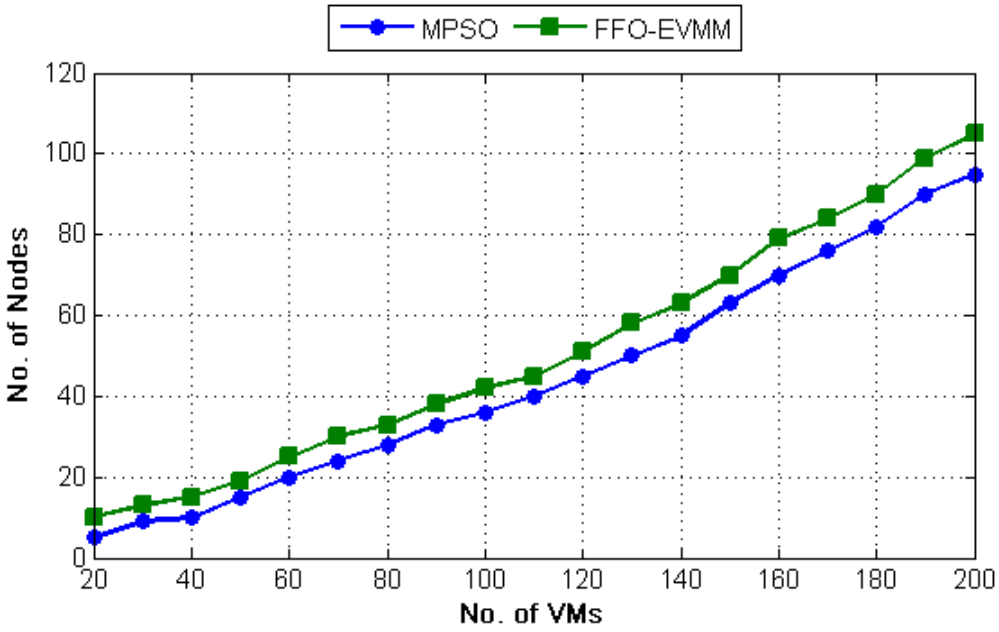


Figure 4.11: VMs vs Nodes

Figure 4.12 portrays the number of VM migrations performed by both FFO-EVMM and MPSO techniques. The results depict that FFO-EVMM performs much less VM

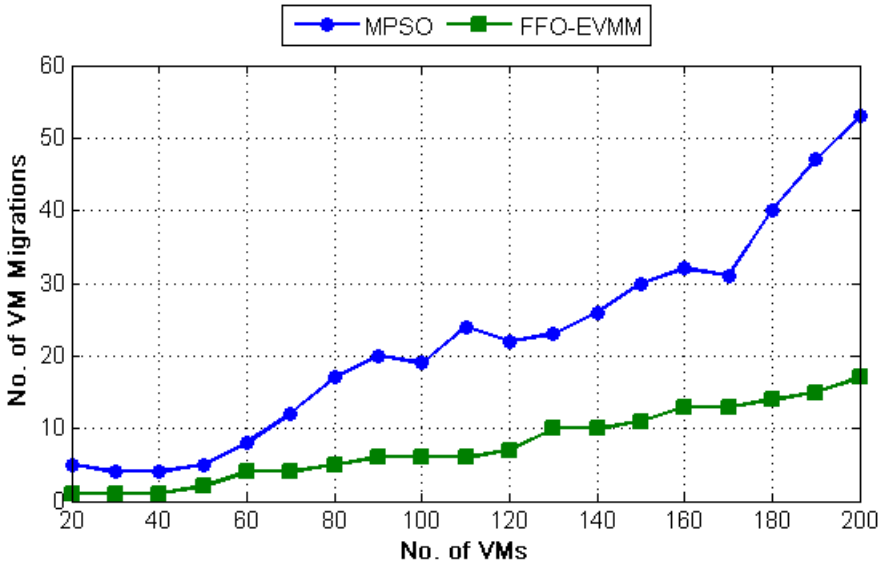


Figure 4.12: VMs vs VM Migrations

migrations than MPSO. Instead of using more number of nodes than MPSO, FFO-

EVMM initiates less VM migrations, as VM migrations are done from the optimized nodes only, which averts unnecessary migrations. Also, the selection of the accurate nodes for VM migration is done by the firefly optimization algorithm at a higher convergence rate than that of PSO. The nodes are selected with reduced discovery time resulting in an optimal node utilization. The node utilization refinement minimizes the number of VM migrations thereby saving energy.

Figure 4.13 gives the graph for the energy consumed by both the techniques. As observed from the figure, FFO-EVMM consumes less energy than MPSO. This is due to the reason that FFO-EVMM uses much less VM migrations than MPSO, leading to the energy saving. Also, it locates and diminishes the number of active nodes to reduce consumed energy. It improves the level of node utilization, keeping in view the resource contention, thereby influencing VM migrations. And since the VM migrations are performed from the optimized nodes only, unnecessary migrations are avoided resulting in cutting down of the consumed energy caused due to these migrations, hence accomplishing energy efficiency.

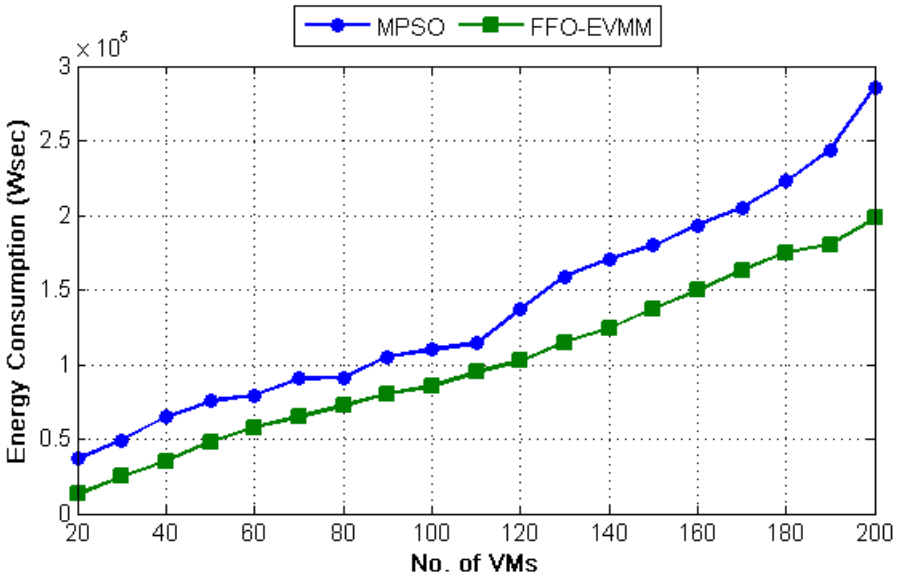


Figure 4.13: VMs vs Energy Consumption

4.4.1 Discussion

Figure 4.14 substantiates the potentiality of FFO-EVMM over MPSO technique. The results clearly justify that an average of 66.25% VM migrations and 29.49% of energy has been conserved using FFO-EVMM over MPSO technique. Though MPSO technique has outperformed FFO-EVMM in terms of number of nodes saved by saving an

average of 17.13% nodes, but the saved VM migrations and energy saved are far better in FFO-EVMM than MPSO.

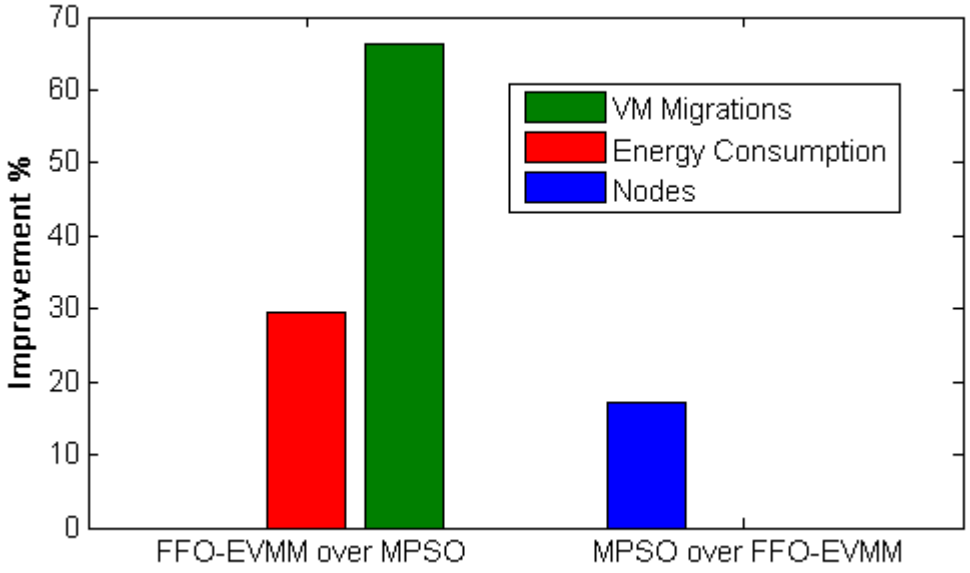


Figure 4.14: Improvement Graph of FFO-EVMM & MPSO

4.5 Conclusion

This chapter discussed the detailed design of the proposed FFO-EVMM technique along with its algorithm and flowchart. The chapter also depicted the performance analysis of the proposed FFO-EVMM technique using CloudSim simulator. The outcomes determined that an average of 72.34% of migrations have been reduced and 34.36% of nodes have been saved with FFO-EVMM. Due to the less number of migrations and nodes, an average of 44.39% of energy has been saved using FFO-EVMM over ACO-based and FFD-based techniques. This chapter further presented the validation of FFO-EVMM with an existing MPSO approach. The results depicted that an average of 66.25% VM migrations and 29.49% of energy have been conserved using FFO-EVMM over MPSO technique

The next chapter discusses the design of the proposed Energy-aware Load Balancing (ELB) model and the two proposed ELB techniques for cloud computing.

Chapter 5

Proposed Energy-aware Load Balancing Techniques

The previous chapter discussed in detail the design of the proposed FFO-EVMM technique that performs live migration of most-loaded VMs to the most energy-aware nodes. It maximizes energy-efficiency through optimum VM migrations, thereby improving resource utilization and sustaining scalability to a large number of cloud nodes.

In this chapter, an Energy-aware Load Balancing (ELB) model, that makes ELB decisions from both the provider and the user perspective, has been proposed. Also, two ELB techniques with and without resource utilization have been proposed, namely ELB(RU) and ELB(w/o RU). ELB(RU) uses an amalgamation of ERU and FFO-EVMM approaches, whereas ELB(w/o RU) uses only FFO-EVMM algorithm. Both the techniques aspire to balance the load of cloud data centers, while trying to maximize energy efficiency through efficient usage of cloud resources and enhanced performance. By reducing energy consumption, these techniques indirectly tend to reduce carbon emissions and cooling requirements of cloud data centers to achieve green computing.

This chapter initially presents the ELB model along with its modules, working phases, design of ELB(RU) & ELB(w/o RU) techniques and detailed description of their proposed algorithms. Finally, the performance analysis of ELB techniques over an existing approach and later, a comparison based validation table has been presented.

5.1 Proposed Energy-aware Load Balancing (ELB) Model

The following section describes the proposed Energy-aware Load Balancing (ELB) Model with its features, modules and functioning. The ELB model assists the system administrators to balance the load of the system in an energy-aware manner. It endeavors to maximize the utilization of the system resources thus improving the energy efficiency of the system by appropriately assigning the users workloads to the energy-aware nodes. Additionally, by using the migration of VMs, containing the workloads, to the more energy-efficient nodes, it tends to further minimize the systems energy needs and balance the system workload, thereby achieving the load balancing. Further, it attempts to enhance the system's performance and attains green computing without degrading the performance of the system. For taking energy-aware load balancing decisions, the model uses two energy-aware load balancing (ELB) techniques which will be described later in this section. The objective of the ELB model is to make apposite energy-aware load balancing verdicts. The key features of the proposed ELB model are listed below.

- CPU and memory, being mainly accountable for increased energy consumption in a node, ELB model tries to optimize their consumed energy.
- Energy consumed by each node is measured in terms of power consumed by CPU and memory over a period of time.
- Local Energy Calculator unit is deployed on each node to keep a track of the energy consumed by each node in the cloud data centre.
- The ELB model uses two energy-aware load balancing (ELB) techniques to make energy-aware load balancing decisions.
- This ELB model running ELB techniques, uses a scheduler to act according to the two techniques. This scheduler acts as ABC scheduler for ELB(RU) technique, whereas it works as a simple scheduler for ELB(w/o RU) technique.
- It also uses an energy-aware VM migrator running an energy-aware VM migration technique.
- The ABC scheduler acts as an energy-aware scheduler by running the ABC-based ERU (Chapter 3) and schedules the workload to the energy-aware resources according to the energy constraints thereby offering faster executions and green advantage.

- The energy-aware VM migrator running an energy-aware VM migration technique (Chapter 4) migrates the VMs to the more energy-efficient nodes to balance the system's load and to reduce the energy consumption of the data center, thereby achieving the green computing.

- A Workload DataBase (WDB) is maintained to store the past resource utilization and the energy consumption information to help in taking the future energy-aware resource allocation decisions. Figure. 5.1 shows the ELB model.

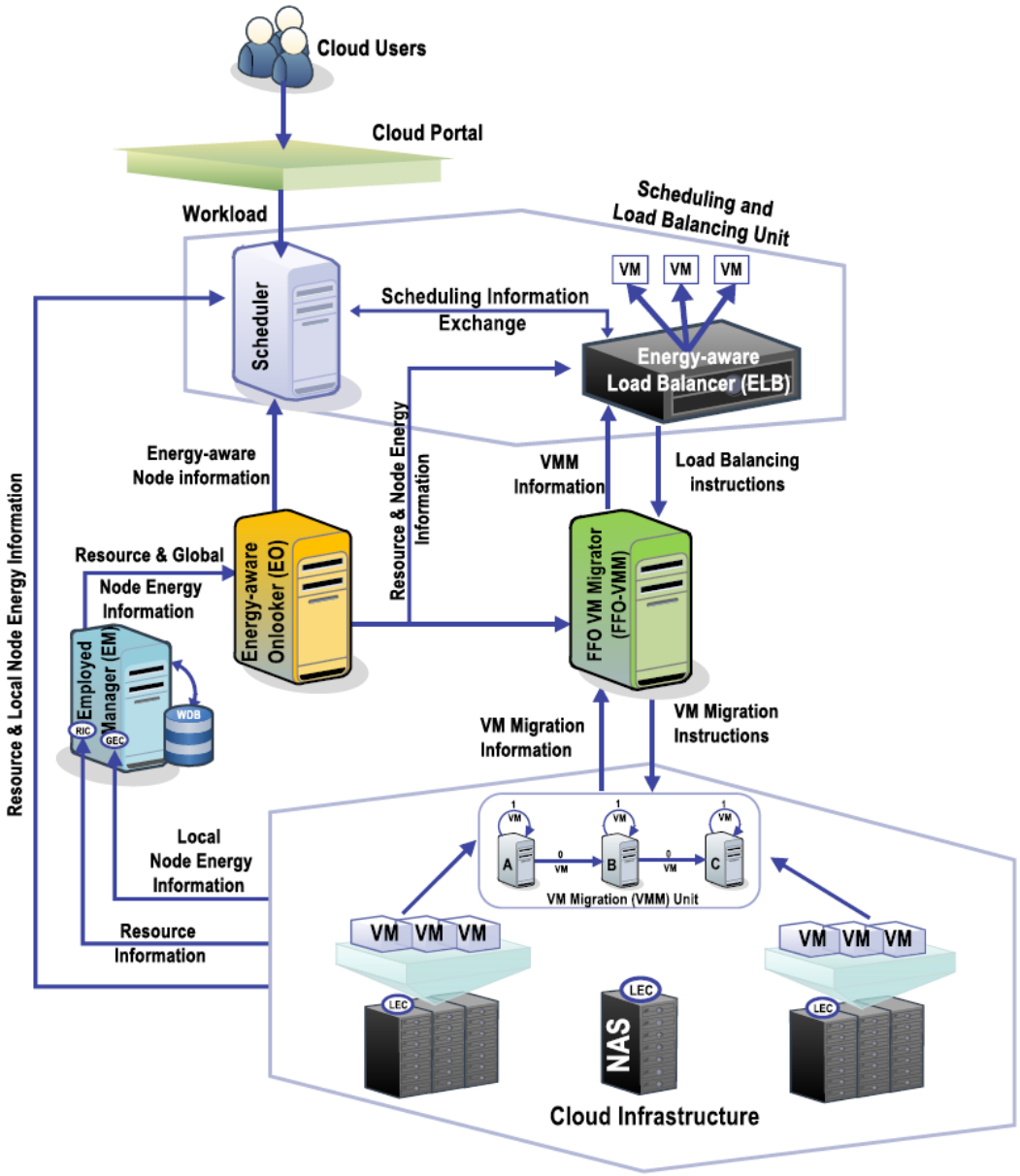


Figure 5.1: Energy-aware Load Balancing Model

5.1.1 Modules of ELB Model

This subsection describes and discusses the various modules of the proposed ELB model. These are as follows :

- Cloud Portal : The cloud portal assists the cloud users to submit the workloads to the cloud. Each cloud user submits a workload (set of jobs) to the cloud along with the required number of CPUs, required amount of memory, the type of the workload (whether CPU-intensive or memory-intensive) and the preference of cost/performance.

- Employed Manager (EM) : The EM is one of the modules working for ABCS. This module is further comprised of RIC and GEC sub-modules that help EM to gather the resource and energy related information about each node in the system. The EM further provides this information to the EO and helps it in taking an energy-aware decision.

- Resource Information Collector (RIC) : The RIC is one of the two parts of the EM module that collects all the information about the resources (including the CPU, the memory and the VMs) on all the nodes in the system. It then updates the EM with its collected information.

- Global Energy Collector (GEC) : It is the other part of the EM module that collects the energy-consumption information of all the nodes from the LECs deployed on each node in the system. The EM is further updated with this energy-related information.

- Local Energy Calculator (LEC) : This local energy collector is positioned on each node to calculate its consumed energy. All the LECs then provide their information to the GEC sub-module of the EM module.

- Network Attached Storage (NAS) : A central storage is provided to all the files in the system by using NAS to allow more hard-disk storage space and ease of access.

- Workload DataBase (WDB) : It is a repository to store the data about the resources and the energy used by the workloads in the past.

- Scheduler : This scheduler works, both as an ABC scheduler and the simple scheduler. The ABC scheduler schedules the workloads to the required resources by using ABC-based ERU, which works for the ELB technique with resource utilization

(ELB(RU)). The simple scheduler schedules the workloads for the other ELB technique without resource utilization (ELB(w/o RU)) . The scheduling information is further delivered to the energy-aware load balancer (ELB).

- Energy-aware Onlooker (EO) : It gathers the resource and energy related information from the EM and makes a decision on the best energy-aware node for the workload. This information is then provided to the scheduler. The resource and energy related information is also provided to the FFO-VMM and the ELB modules.

- Cloud Users : The cloud users submit their workloads to the scheduler and get the required resources in return to execute their workloads.

- Cloud Infrastructure : It is composed of the actual physical server nodes with the running Virtual Machines. One of the server nodes acts as a Network Attached Storage to provide a central storage for the whole data. The nodes are set aside in energy-saving mode to save energy and are brought to the active mode as and when required.

- FireFly Optimization-Virtual Machine Migrator (FFO-VMM) : After receiving the resource & the energy related information from the EO and the load balancing instructions from the ELB, this module activates the virtual machine migration (VMM) unit by sending the VM migration instructions to it. An energy-aware virtual machine migration technique based on the firefly optimization is used to decide the VMs to be migrated, the source from where these VMs will be migrated and the destination where these VMs will be migrated. Also, the ELB module is further updated with the virtual machine migration (VMM) information.

- Virtual Machine Migration (VMM) Unit : This unit is the part of cloud infrastructure and performs actual VM migrations after receiving the VM migration instructions from FFO-VMM module. Here, 1 indicates the migration of a virtual machine whereas 0 indicates that no virtual machine has been migrated. After the migration, FFO-VMM module is updated with VM migration information.

- Energy-aware Load Balancer (ELB) : This is the core module of the ELB Model. It runs two energy-aware load balancing (ELB) techniques to make energy-aware load balancing decisions. One ELB technique considers resource utilization and the other ELB technique considers meeting a high performance level rather than the utilization of the resources. It receives the scheduling information from the scheduler, the resource and the node energy information from the EO module and

the VM migration related information from the FFO-VMM module. After collecting this, ELB balances the workloads of the system nodes by making an energy-aware migration decision without degrading their performance.

5.1.2 Phases of ELB Model

The ELB model works through the two different scheduling phases that uphold the overall operation of balancing the system's load across various servers in an energy-aware manner. Further, it has the capability to utilize the resources of the data center in an efficient way, thereby dropping the levels of consumed energy in the data center. The ELB model processes variable workloads & supports varied nodes, having different computing capabilities and possessing variable power consumption values, depending on their operational computing units. In this model, the scheduler acts, both according to the cloud provider and the cloud user. As the cloud provider always wants to utilize resources efficiently to save energy in cloud data centers, therefore, the scheduler runs ELB(RU) technique to enhance resource utilization. And, the cloud user always wants to process more numbers of workloads in less time, therefore, the scheduler runs ELB(w/o RU) technique to enhance the performance. Both the techniques try to save energy in the cloud data centers, thereby reducing carbon emissions and cooling requirements to achieve green computing. The different scheduling phases of ELB model are as follows:

5.1.2.1 Scheduling phase I

In the first scheduling phase, the system resources are utilized optimally to save energy without degrading the system's performance. For this, the ABC optimization based ERU (as discussed in Chapter 3) has been used.

Whenever a workload is submitted by the cloud user along with the preference of cost/performance, it is directed to the scheduler which acts according to the preference entered by the user. If the user enters the cost as a preference, the scheduler runs the ELB(RU) technique to lower the number of used nodes by optimally utilizing the resources, thereby averting the energy consumption and hence the cost. Whereas, if the user enters the performance as a preference, the scheduler run the ELB(w/o RU) technique to process maximum number of workloads thereby enhancing the performance.

The scheduler acts as the ABC scheduler for the proposed ELB(RU) technique. The ABC scheduler schedules the workload to the appropriate energy-aware ingredient

resources by the above-stated ERU technique, depending on the information received from EO. EO gathers this resource and energy related information from EM and makes a decision on the best energy-aware node for the workload. EM module together with its sub-modules, RIC & GEC, collects the resource and energy related information about each node in the system. To decide about the suitable energy-aware node for the workload, EO keeps a check on the available resources & consumed energy of all the nodes by specifying certain thresholds. A node in an energy-saving state, is switched 'on', only if the present active nodes are not sufficient to meet the requirements of the workload. The VMs are started to process the workloads and stopped to save energy. The entire resource utilization and energy consumption data of the processed workloads are saved in the WDB to be utilized for the future use. The resource and energy related information is also provided to FFO-VMM and ELB modules to make a judgement on the second phase of scheduling.

In the same way, the scheduler acts as a simple scheduler for the ELB(w/o RU) technique that simply schedules the workloads to the available resources without checking their utilization, to process more number of workloads. Thereafter, when the energy consumed by any node, processing the workloads, start increasing beyond a threshold, this information is provided to FFO-VMM and ELB modules to be ready for the second phase of scheduling.

5.1.2.2 Scheduling phase II

In the second phase of the scheduling, the system load is balanced on the least requisite nodes by migrating VMs. This is done in a way to improve resource utilization, thereby enhancing energy efficiency of the cloud data center. A firefly optimization based energy-aware VM migration technique (FFO-EVMM)(as discussed in Chapter 4) has been used to effectively migrate the VMs to the more energy-efficient nodes to balance the system's load and to cut down the energy demand. VM migration is required, when at run-time, the energy consumption of a node increases beyond a threshold.

The main module of the ELB model is the ELB module which runs two ELB techniques to draw an ELB decision. In case of ELB(RU), after collecting the scheduling information from ABC scheduler, the resource and the node energy information from the EO module, the ELB takes an energy-aware migration decision and directs this information to FFO-VMM module. Upon receiving the resource & the energy related information from EM and the load balancing instructions from ELB, FFO-VMM module initiates VMM unit. FFO-VMM module uses the above-mentioned FFO-EVMM

technique to perform required VM migrations. VMM unit executes the actual VM migrations after receiving VM migration instructions from FFO-VMM. The migration of a VM is indicated by 1 and 0 otherwise. After the migration, FFO-VMM & ELB modules are updated with VM migration information. Whereas, in case of ELB(w/o RU), the scheduling information collected only from the scheduler is passed onto ELB module to decide about energy-aware migration for load balancing. This information is then given to FFO-VMM module which initiates VMM unit to perform required VM migrations. By this way, the load of the system is balanced in an energy-efficient way, therefore boosting the levels of resource utilization, enhancing the performance levels and reducing energy usage.

After an overview of the key characteristics, the various modules, the working phases & the data flow representation of the proposed ELB model, the following section provides the design of the proposed ELB technique with resource utilization along with the detailed description of the proposed algorithm.

5.2 Energy-aware Load Balancing Technique with Resource Utilization - ELB(RU)

This section proposes an Energy-aware Load Balancing (ELB) technique with resource utilization for a cloud environment. This technique considers the utilization of the resources to lower the consumed energy in the cloud data centers without degrading the system's performance. The prior work related to the resource utilization and the migration aspects of the proposed technique has already been discussed in Chapter 3 and Chapter 4.

The proposed ELB(RU) technique is an amalgamation of the two approaches, one of which is ABC optimization based ERU, that aspires to maximize the energy-efficiency through the best usage of the resources. The other technique is FFO optimization based FFO-EVMM technique, that attempts to maximize the energy-efficiency through the optimum migration of VMs, thereby improving the resource utilization levels.

Overall, the ELB(RU) technique with the help of ERU & FFO-EVMM aspires to balance the system load thereby achieving load balancing. This is done in a way to enhance the utilization levels of CPU & memory thereby reducing the number of active servers & the number of VM migrations, hence achieving a hike in the energy-efficiency of the cloud data centers without diminishing its performance.

The problem formulation, energy model and the system model are already been explicated in Chapter 3. Algorithm 3 represents the pseudocode for the proposed ELB(RU) technique, which uses two functions, namely ABC_Scheduler() & FFO_VM_Migrator().

Algorithm 3: ELB (RU) Algorithm

```
Data: set of jobs, set of nodes
Result: Finding the best job-node pair
begin
    JOB ← set of jobs
    R ← set of nodes
    PREU ← boolean variable        // 1- If past resource and energy utilization data is present in WDB
    C = 1                          // Number of cycles for improvement phase
    InputMCN, M, W_Type, Avail_CPU, Avail_Mem, CPU_req, Mem_req,
    Energy_req, EC, CPU_th, Mem_th, Energy_th
    for (each job ∈ JOB) do
        if (PREU = 0) then
            // If past resource and energy utilization data not present, send jobs directly to scheduler
            ABC_Scheduler()
        else
                    // Factorize the workload as cpu-intensive (CW) or memory-intensive (MW)
            if (W_Type = CW) then
                Assign_to_CPU()
            else
                Assign_to_Memory()
            ABC_Scheduler()                          // Call to ABC Scheduler
        Repeat until all the jobs are assigned to nodes
    // If Energy consumption (EC) of a node is greater than the energy threshold, then VMs are migrated to avoid hot-spots and to balance the load
    for (each node) do
        if (EC > Energy_th) then
            FFO_VM_Migrator()                        // Call to FFO-VM Migrator
        Repeat till the load is balanced
```

Algorithm 4 represents the pseudocode for the ABC_Scheduler() function, whereas Algorithm 5 represents the pseudocode for the FFO_VM_Migrator() function respectively. Figure 5.2 depicts the flowchart for the proposed ELB (RU) technique that uses ERU & FFO-EVMM to balance the load and to reduce the energy consumption of the cloud data centers.

Algorithm 4: ABC_Scheduler()[Chapter 3]

begin
- Initialization_phase()
- **begin**
  - $(\vec{y_m})s \leftarrow$ population of all possible solutions // All possible solutions are initialized
  - $y_{mi} = l_i + rand(0,1)(u_i - l_i)$
- **while** *(C < MCN)* **do**
  - Employed_Bees_Phase()
  - **begin**
    - $\upsilon_{mi} = y_{mi} + \phi_{mi}(y_{mi} - y_{ki})$ // Produce New Solution
    - // Memorize the Better Solution
    - **if** *($\vec{\upsilon_m} > \vec{y_m}$)* **then**
      - memorize $(\vec{\upsilon_m})$
    - **else**
      - keep $(\vec{y_m})$
  - Onlooker_Bees_Phase()
  - **begin**
    - **if** *($CPU_{req} <= Avail\_CPU$) && ($Mem_{req} <= Avail\_Mem$)* **then**
      - **if** *($CPU_{req} < CPU_{th}$) && ($Mem_{req} < Mem_{th}$)* **then**
        - **if** *($Energy_{req} < Energy_{th}$)* **then**
          - $p_{mi} = \frac{fit_{mi}(\vec{y_m})}{\sum_{m=1}^{M} fit_m(\vec{y_m})}$ // Calculate Probability
          - $\upsilon_{mi} = y_{mi} + \phi_{mi}(y_{mi} - y_{ki})$ // Produce New Solution
          - // Memorize the Better Solution
          - **if** *($\vec{\upsilon_m} > \vec{y_m}$)* **then**
            - memorize $(\vec{\upsilon_m})$
          - **else**
            - keep $(\vec{y_m})$
  - Scout_Bees_Phase()
  - **begin**
    - $y_{mi} = l_i + rand(0,1)(u_i - l_i)$ // Determine Abandon Solution
  - Memorize Best Solution
  - Increment C by 1
- Output the best solution achieved

Algorithm 5: FFO_VM_Migrator()[Chapter 4]

begin

Source_Node()

for *(each Node)* **do**

Compute Energy Consumption using

$$EC_i = (\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times t \tag{5.1}$$

$EC[] \leftarrow$ Energy Consumption value

Calculate Execution Time using

$$ET_i = \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} ET_{ijk} \tag{5.2}$$

$ET[] \leftarrow$ Execution Time value

for *(each Node)* **do**

Compute Attraction Index, $AI_i(EC_i, ET_i)$

$AI[] \leftarrow$ Attraction Index value

Sort $AI[]$ in an ascending order according to Energy Consumption values

Compute Distance using

$$Distance = Avg(AI_{mid}, AI_{max}) \tag{5.3}$$

Find the node with Energy Consumption value nearest to the calculated Distance value from the sorted $AI[]$

Culprit_VM()

for *(each VM on the Source Node)* **do**

Compute Load using

$$Load_{ij} = \frac{\sum_{j=1}^{n_i} job_{ij}}{(\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times \Delta t} \tag{5.4}$$

$Load[] \leftarrow$ Load value

Sort $Load[]$ in a descending order

Destination_Node()

Get the first element of sorted $AI[]$ as the destination node

Move the first element of $Load[]$ to the first element of $AI[]$

Update the Distance value using

$$(Distance)^{t+1} = (Distance)^t + \frac{\sum_{j=1}^{n_i} job_{ij}}{(\sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rpu_{ijk} + \sum_{j=1}^{n_i}\sum_{k=1}^{l_i} rmu_{ijk}) \times \Delta t} + \epsilon \tag{5.5}$$

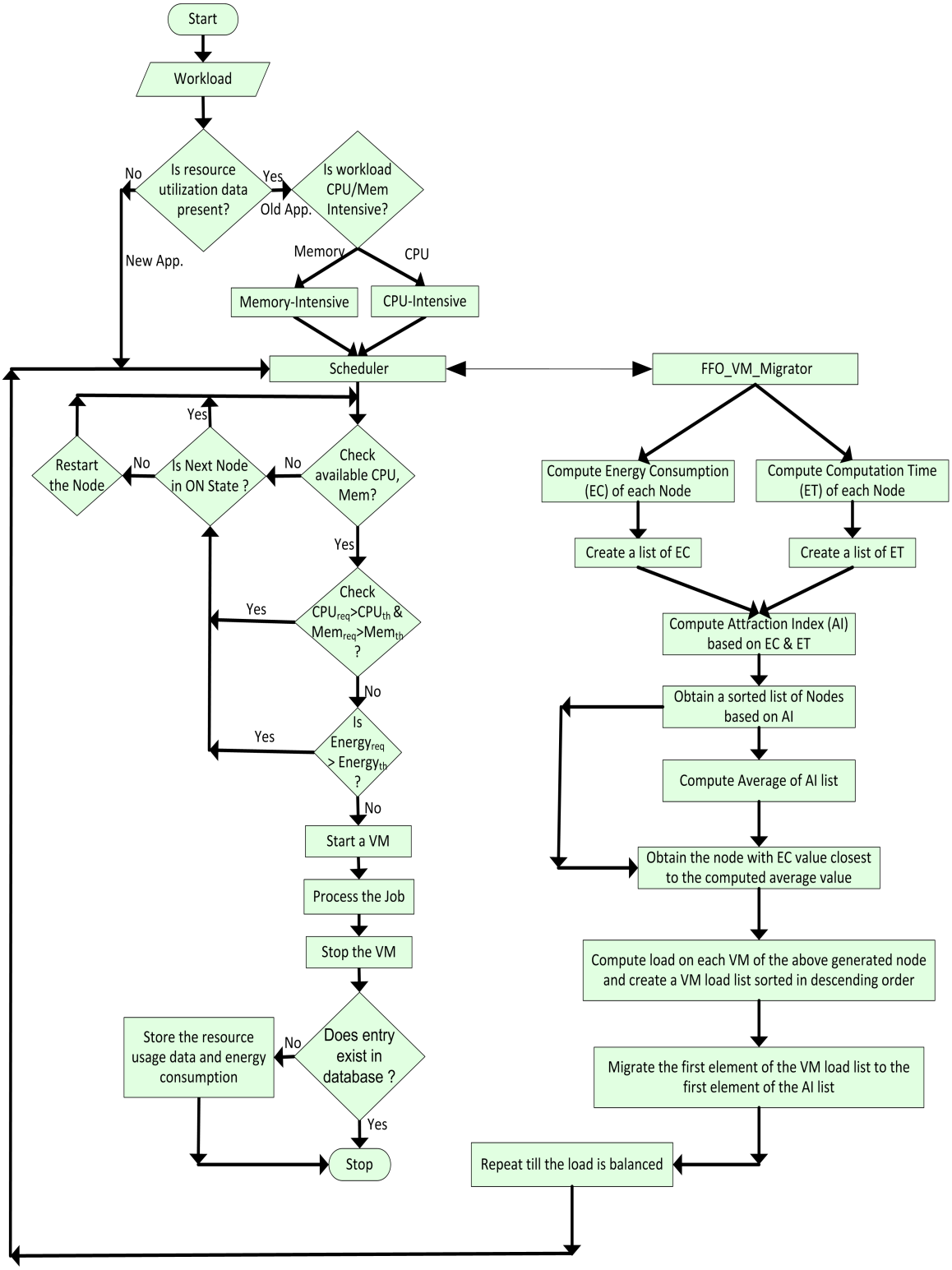


Figure 5.2: Flowchart for ELB(RU) Technique

After giving the details of the proposed ELB technique with resource utilization, the following section provides the design of the proposed ELB technique without resource utilization (ELB(w/o RU)) along with its detailed description and the algorithm.

5.3 Energy-aware Load Balancing Technique without Resource Utilization - ELB(w/o RU)

This section proposes another Energy-aware Load Balancing technique for the cloud computing environment, namely ELB(w/o RU), that does not consider the resource utilization at its priority. Its main criterion is to achieve a higher user satisfaction level which can be obtained by handling more number of workloads received from the cloud users. It assigns the workload to the first available node in the system fulfilling its resource requirements. This helps in enhancing the performance as workloads are executed without searching for the best resources. So, the more the number of workloads handled, the higher is the user satisfaction level and so is the performance. Also, it tries to abate the consumed energy in the cloud data centers by migrating the VMs from the less energy efficient nodes to the more energy efficient nodes, without debasing the system's performance. For this, the above-mentioned FFO-EVMM is used that strives

Algorithm 6: ELB (w/o RU) Algorithm

begin

Step 1. Input workloads and nodes with resource details.

Step 2. If their past resource and energy utilization data is not present, send workloads directly to scheduler.

Step 3. Or Factorize workloads as cpu-intensive (CW) or memory-intensive (MW) and then send them to scheduler.

Step 4. Scheduler schedules workloads to appropriate resources

Step 5. Repeat Step 2 to Step 4 until all workloads are assigned to appropriate nodes by scheduler.

Step 6. Check Energy consumption (EC) of each node. If it is greater than energy threshold of that node, then perform Step 7.

Step 7. Call FFO_VM_Migrator() to migrate the VMs to avoid hot-spots and to balance the load.

Step 8. Repeat Step 6 & Step 7 till the load is balanced.

to augment the energy-efficiency through the optimum migration of VMs, thereby accomplishing load balancing. Overall, this ELB(w/o RU) technique with the help of FFO-EVMM technique, aims to attain load balancing through enhanced performance, hence acquiring a boost in the energy-efficiency of cloud data centers.

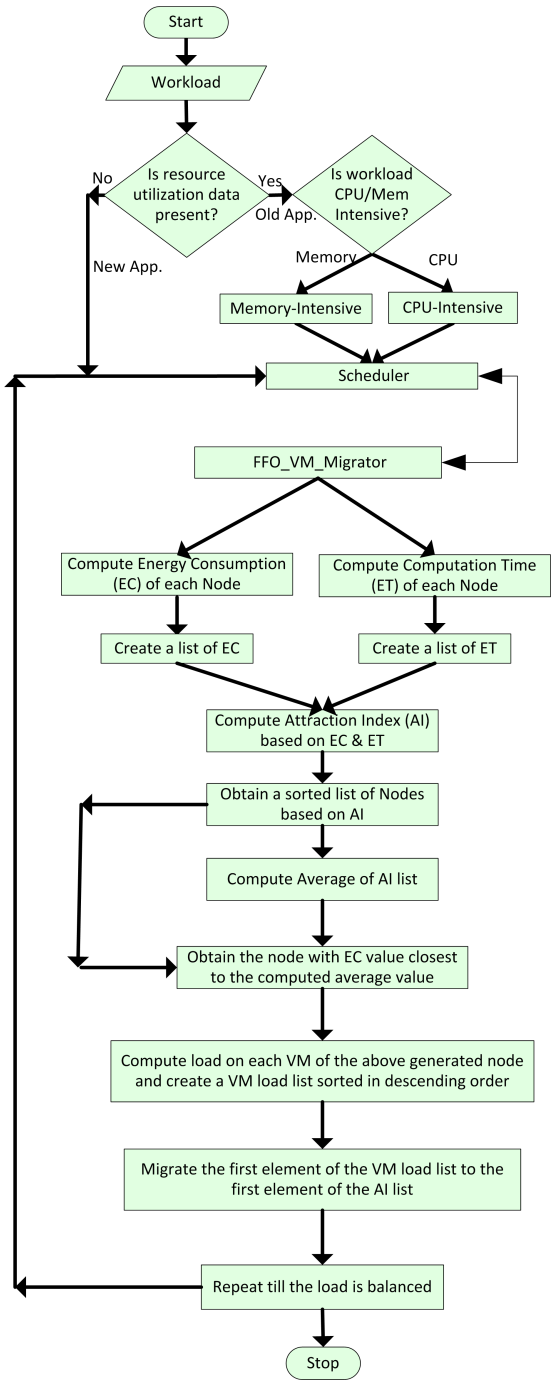


Figure 5.3: Flowchart for ELB(w/o RU) Technique

Algorithm 6 represents the pseudocode and Figure 5.3 depicts the flowchart for the proposed ELB (w/o RU) technique that uses the FFO-EVMM technique to migrate the VMs in order to balance the load and to curtail the energy consumption of cloud data centers.

This section projected the detailed design, algorithm and the flowchart of the proposed ELB techniques. The next section portrays the performance analysis of the proposed ELB techniques by using CloudSim simulator followed by a detailed discussion on the effectiveness of the proposed techniques over the existing approach.

5.4 Experimental Results of ELB Techniques

This section evaluates the proposed ELB algorithms and compares them with the existing Enhanced Load Balancing (EnhancedLB) algorithm [206], using the CloudSim toolkit [200]. The details of the CloudSim toolkit has been given in Chapter 3. The EnhancedLB is a load balancing approach that uses the static value of the upper and the lower threshold to find the overloading and the underloading situation of a node and then selects the node for migrating the VM. Thereafter, it places the selected VM to the minimum power consuming node, thereby minimizing the energy consumption [206].

5.4.1 Performance Evaluation

Up to 50 VMs and 50 nodes have been simulated and the simulation is repeated for multiple runs. The simulations have been carried out by using the same evaluation parameters as defined by Jain et al. [206]. Table 5.1 & Table 5.2 give the specification details of a node and a VM respectively.

Table 5.1: Node Description

Parameter	Value	Comment
Node_MIPS	1000-3000	MIPS allocated to each Node
Node_RAM	10000MB	Primary Memory allocated to each Node
Node_BW	100000 bits/sec	BandWidth allocated to each Node

Table 5.2: VM Description

Parameter	Value	Comment
VM_MIPS	250-1000	MIPS allocated to each VM
VM_RAM	128-1024 MB	Primary Memory allocated to each VM
VM_BW	2500-7500 bits/sec	BandWidth allocated to each VM

The number of VM migrations and the energy consumption values have been calculated through all the three techniques. The obtained results have then been compared as shown in Figure 5.4 & Figure 5.5.

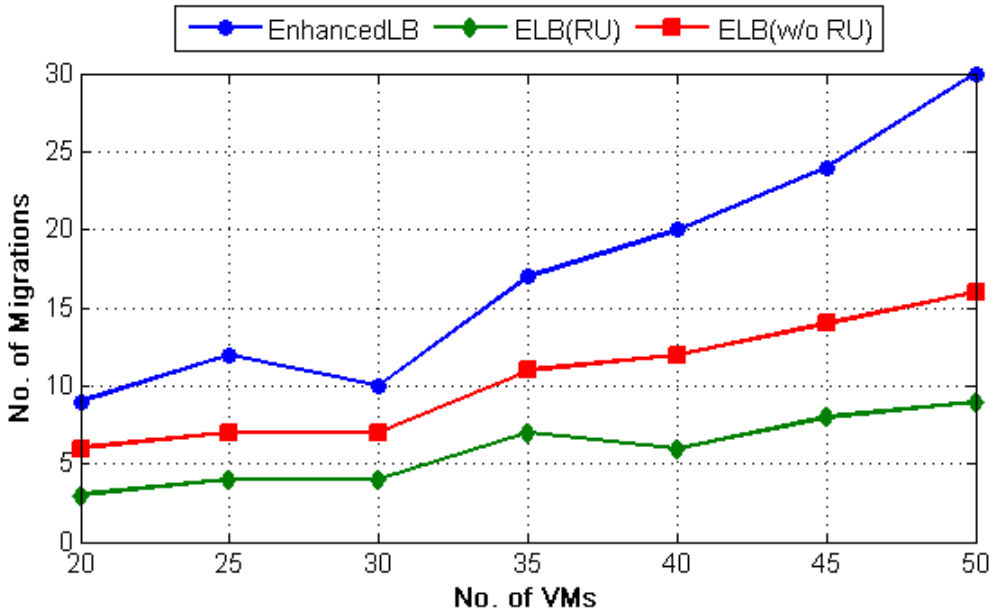


Figure 5.4: VMs vs VM Migrations

Figure 5.4 presents the comparative view of the proposed ELB(RU) and ELB(w/o RU) techniques with the EnhancedLB approach on the basis of the required number of VM migrations over the number of VMs as independent axis. It is important to place the VMs carefully on the nodes to enhance resource utilization and avoid unnecessary VM migrations, thereby enhancing energy efficiency. The resource contention should also be kept in mind while allocating the VMs to the nodes as the resource contention may lead to the performance degradation.

Based on the analysis of the obtained results, it is evident that ELB(RU) performs lesser number of VM migrations in comparison to the EnhancedLB approach. This is because ELB(RU) runs ERU technique for placing VMs on the nodes by considering contention of resources and to achieve optimal resource utilization. It then uses FFO-EVMM to migrate VMs from the optimally utilized resources only, to avoid needless VM migrations. The number of VM migrations performed by ELB(w/o RU) is also less than the migrations performed by EnhancedLB. The reduced number of VM migrations in ELB(w/o RU) is due to the use of FFO-EVMM migration technique which optimally migrates the VMs, thereby reducing the number of VM migrations. Whereas the EnhancedLB approach allocates as many VMs as possible on a node leading to

enhanced VM migrations and energy consumption.

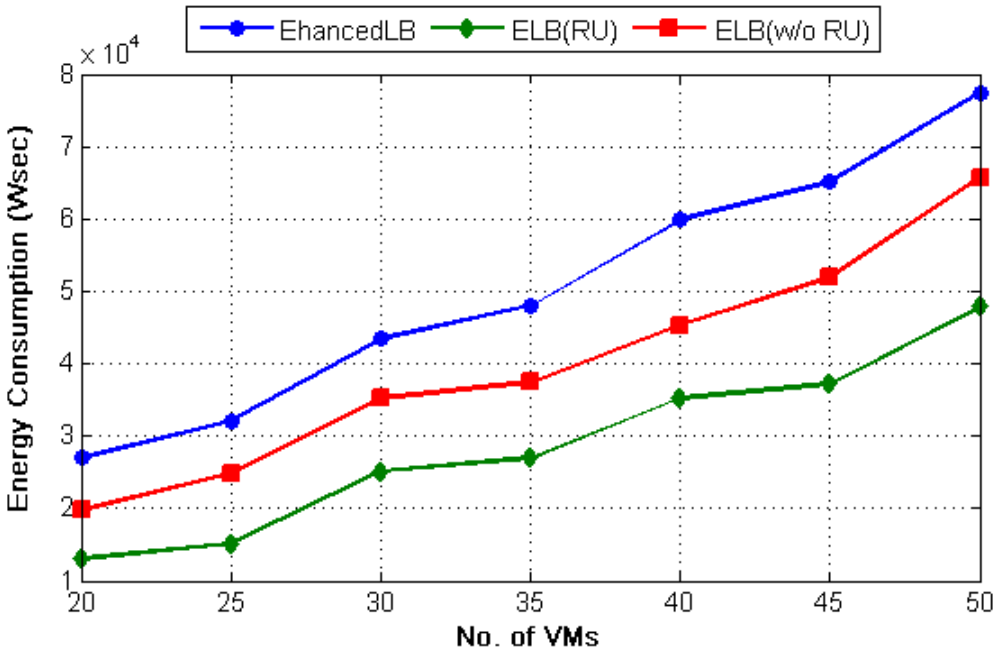


Figure 5.5: VMs vs Energy Consumption

The graph given in Figure 5.5 depicts the energy consumed by each of the three techniques. As can be observed from the figure, ELB(RU) and ELB(w/o RU) both consume less energy as compared to EnhancedLB. The reduction of energy demand in ELB(RU) is due to the lesser number of required VM migrations that have been achieved by the enhanced resource utilization. The reduction in the number of VM migrations brings down the amount of energy consumed, that would have otherwise been misspent when the VMs are being migrated. Consequently, the required operational energy and the energy consumption level drops down. The decrease in the consumed energy of ELB is also due to the improvement attained in the host utility levels because of energy-aware VM migration decisions. Whereas, ELB(w/o RU) consumes less energy due to performing less number of VM migrations. Also, these VM migrations are energy-aware that leads to the reduction in the energy consumption. However due to the maximum number of VM migrations, the energy demand of the EnhancedLB approach is the highest.

5.4.2 Discussion

The efficacy of the proposed ELB techniques has been highlighted in Figure 5.6. The results demonstrate that an average of 65.55% of VM migrations and 44.65% of energy have been conserved using the proposed ELB(RU) approach over the EnhancedLB approach. Whereas, 38.38% migrations and 21.40% energy have been saved using the ELB(w/o RU) approach against the EnhancedLB approach.

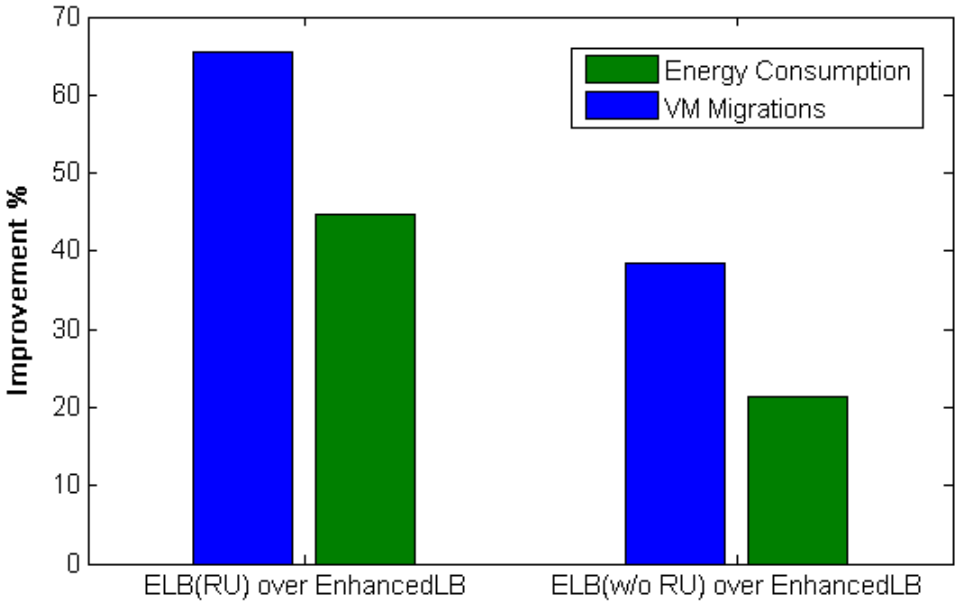


Figure 5.6: Improvement Graph of ELB over EnhancedLB

This section depicted the performance analysis of the proposed ELB techniques over an existing approach. The next section drafts the validation of the proposed ELB techniques over the existing techniques.

5.5 Validation of Proposed ELB Techniques

In the last three years, the techniques mentioned in the below table have proposed the research work related to energy-aware load balancing. Table 5.3 portrays the comparison of the proposed ELB techniques with these existing techniques.

Table 5.3: Validation Table of Proposed ELB Techniques with Existing Techniques

Techniques	Parameters	Approach Used	Optimization Results	Limitations
PLBA [207]	Mean Response Time, Energy Consumption	Utilizes both dynamic voltage/frequency scaling & server consolidation to save energy	Up to 17.2%-35.3% energy saving & 5%-12% reduced response time	Only CPU is considered as a resource
DWOLB (Dynamic well -organized load balancing algorithm) [208]	Cost, Energy consumption, Fitness value, Performance Degradation, Fitness score	Uses Genetic Algorithm to distribute load on VMs for optimum resource consumption & throughput, and to reduce response time	25% energy saving	Cost, resource consumption, throughput & response time are all implicit
Bee-MMT (artificial bee colony algorithm - Minimal migration time) [209]	SLA Violation, Energy Consumption	Uses ABC to detect over and underutilized hosts and then uses MMT to migrate VMs	Up to 26.46% energy saving	Up to 6.45% more SLA Violation
ECLBDVR (Energy Aware Cloud Load Balancing using Dynamic Placement of Virtualized Resources) [210]	Number of running servers, Total Operational Cost, Energy Consumption	Uses virtualization technology to allocate resources on the basis of application demands, Minimizes energy consumption by minim--izing number of servers while avoiding over--loading & overheating	38% reduction in running servers	Operational cost and energy consumption are implicit
Firefly LB [211]	Processor Utilization, Average Response	Uses firefly to balance the load with the objective	10-14% energy saving, 6-7 % less response time, 5-6% better	Only CPU is considered as a resource

	Time, Average Energy Consumption	to maximize resource utilization, minimize response time and energy consumption	processor utilization	
DTLBA (Double Threshold Based Load Balancing Approach) [212]	No. of Migrations, Energy Consumption	Uses Lower threshold to consolidate servers and upper threshold to balance load	Up to 19% reduction in Migrations & 15% energy saving	Main target is to reduce number of VM migrations only
Proposed ELB(RU) Technique	CPU, Memory Utilization, Performance, No. of Nodes, No. of VM migrations, Energy Consumption	Uses ABC-based ERU to utilize resources efficiently and FFO-based VM migration to migrate VMs	-Enhancement in 49.68% of CPU usage, 24.41% memory usage, 7.67% performance -Reduction in 53.21% nodes , 63.10% migrations , 40.47% energy	Does not consider network & storage resources & network-intensive & storage-intensive workloads
Proposed ELB(w/o RU) Technique	Performance, No. of Nodes, No. of VM migrations, Energy Consumption	Its main criterion is to handle more number of workloads to enhance performance. Uses FFO-EVMM to save energy through energy-aware VM migrations	Up to 44.42% enhanced performance, 25% nodes saved, 53.91% reduced migrations, 29.43% energy saved	Resource utilization is not the main priority, Does not consider network & storage resources and network-intensive & storage-intensive workloads

The comparative analysis clearly depicts that the PLBA approach achieves up to 17.2%-35.3% energy saving and 5%-12% reduction in response time. The DWOLB technique saves 25% energy by using GA and the Bee-MMT approach saves up to 26.46% energy by using a combination of ABC and MMT. The ECLBDVR approach reduces 38% running servers by using virtualization whereas the Firefly LB approach

saves 10-14% energy, achieves 6-7% less response time and 5-6% better processor utilization. The DTLBA technique reduces up to 19% migrations thereby saving 15% energy. However, the proposed ELB(RU) technique enhances CPU & memory utilization by 49.68% & 24.41% respectively, saves 53.21% nodes, 63.10% migrations saved thereby saving 40.47% energy. It has also achieved a performance enhancement of 7.67%. Whereas, the proposed ELB(w/o RU) enhances the performance by 44.42% and reduces 25% nodes and 53.91% migrations thereby reducing 29.43% consumption in energy.

5.6 Conclusion

This chapter presented an Energy-aware Load Balancing (ELB) model that makes ELB decisions from the perspectives of both the cloud provider as well as the cloud user. Also, two ELB techniques have been proposed for effectively balancing the system load, thereby increasing resource utilization and performance, hence reducing energy consumption of the cloud data center. The chapter also elaborated the performance analysis of the proposed ELB techniques using CloudSim simulator. The results demonstrate that an average of 65.55% of VM migrations and 44.65% of energy have been conserved using the proposed ELB(RU) approach over the existing EnhancedLB approach. Whereas, 38.38% migrations and 21.40% energy have been saved using the ELB(w/o RU) approach against the EnhancedLB approach. Lastly, a comparison table has been presented to validate the proposed ELB techniques.

The next chapter focuses on the verification and validation of the proposed ELB techniques in the BSNL data center, Chandigarh.

Chapter 6

Case Study : Empirical Evaluation of Load Balancing Techniques at BSNL Data Center

The previous chapter presented the design and implementation details of ELB model for a cloud environment that makes energy-aware load balancing decisions, from both the provider and the user perspectives. It also depicted the design of the proposed ELB techniques with and without resource utilization to balance the system load.

This chapter is targeted towards the evaluation of the proposed ELB model that comprises of the two ELB techniques. These techniques have been verified and validated through the case study conducted at a cloud data center, Bharat Sanchar Nigam Limited (BSNL, Telecom), Chandigarh. Corresponding test cases have been designed for evaluating both these techniques, covering the test cases for resource utilization (both CPU and memory), number of nodes used and migrations performed, the amount of energy consumed etc.

At the outset, this chapter outlines the details about the BSNL data center. Then, the proposed ELB(RU) & ELB(w/o RU) techniques have been validated through its case study and the obtained experimental results have been discussed from various perspectives. Finally, the chapter presents the statistical analysis of the obtained results.

6.1 About BSNL Data Center

This section presents a case study of a data center of a Telecom Organization, Bharat Sanchar Nigam Limited (BSNL). BSNL is India's oldest communication service provider and has a customer base of approximately 100 million. It is a government enterprise headquartered in New Delhi and is the largest provider of fixed telephony and broadband services in India. The experiment was conducted at BSNL (North Zone GSM Billing Data center, Chandigarh). The picture of BSNL data center is as shown in Figure 6.1.

Figure 6.1: BSNL Data Center

The Billing and Customer Care System (B&CCS) is an integrated customer care, billing and accounting platform that supports flexible billing for wide range of GSM services, viz. Tele-services, Bearer services, Supplementary services, GPRS, WAP and IN services etc. The major functions of B&CCS are:

1. Inventory and SIM management
2. Activation/deactivation of mobile numbers
3. Provisioning/de-provisioning of various services to the subscribers
4. Handling of requests from customer care centers regarding telephony services and billing queries
5. Swapping of SIM and MSISDN
6. Collecting, processing and storing of CDRs (Call Detail Records) from the Network Elements
7. Rating the CDRs and Billing
8. Payment and Collection
9. Billing for Inter connect Usage Charges
10. Provision for testing new products/services before commercial launching
11. Threshold monitoring
12. Intrusion detection and Fire wall functions
13. Trouble Ticket (Remedy) system
14. Handling of TAPIN & TAPOUT for Revenue settlement with roaming partners

Kenan Order Management, also known as Kenan OM, is a software product that is used as a Customer Relationship Management (CRM) tool. It is also used for order

management (Every deliverable service to the customer e.g. Plan, STD, ISD etc. is provided through orders). A Kenan OM Module consists of one Admin server for configuration of plans, rates, inventory etc. and one Catalog server for maintaining the inventory and mapping of external ids (like MSISDN, IMSI etc. used by network elements, HLR,SMSC etc.) & internal ids (like account number, subscriber number and service like GSM postpaid, GSM prepaid and CDMA prepaid etc.).

As shown in Figure 6.2, There are 1 to N number of customer servers where customer information, plans subscribed by them, bills and call records are maintained. There is one Middleware that is used to integrate Admin, Catalog and Customer servers. Admin, catalog and Customer Servers do not communicate with each other directly, but interact using Middleware API calls. Middleware uses dedicated processes (for doing any job on these servers e.g. getting account information, getting customer server to be used for new account creation, getting inventory information, assignment of inventory etc.) defined in the Middleware configuration file and these processes are spawned at runtime according to the load requirement.

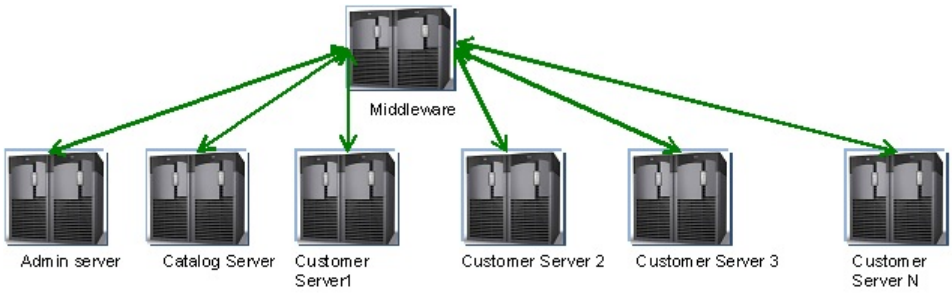


Figure 6.2: Servers in BSNL Data Center

Table 6.1 gives the system configuration of the data center and Table 6.2 gives the details of the abbreviations used in the system.

Table 6.1: System Configuration

Parameter	Value	Comment
Middleware	1	To integrate Admin, Catalog and Customer servers
Admin Server	1	For configuration of plans, rates, inventory etc.
Catalogue Server	1	To maintain the Inventory and mapping of External & Internal Ids
Customer Servers	1-N	To maintain customers' information

There are 19 customer servers, out of which 1 is dedicated to WiMax, 4 to CDMA and 14 are the GSM servers. Out of these 14 GSM servers, 6 are, the GSM postpaid

Table 6.2: Abbreviations Used

B&CCS	Billing and Customer Care System
GSM	Global System for Mobile communication
CDMA	Code Division Multiple Access
WiMax	Worldwide Interoperability for Microwave Access
TRAI	Telecom Regulatory Authority of India
CSR	Customer Service Representative
MSISDN	Mobile Station International Subscriber Directory Number
IMSI	International Mobile Subscriber Identity
SIM	Subscriber Identity Module
HLR	Home Location Register
SMSC	Short Message Service Center
SDP	Service Delivery Platform
API	Application Program Interface
APN	Access Point Name
MNP	Mobile Number Portability
HLR	Home Location Register
SDP	Service Delivery Platform

servers and 8 are the GSM prepaid servers. Whenever there is a request for Postpaid, WiMax and CDMA numbers, these are created on their respective customer servers by Customer Service Representative (CSR). Whereas, prepaid customers are created in batches as more than 95% subscribers are prepaid subscribers. Batch process distributes the numbers on the 8 GSM prepaid customer servers in a round robin fashion. Table 6.3 gives the description of each server, with its hardware configuration and Table 6.4 presents the software configuration of each server.

Table 6.3: Server Description

Parameter	**Value**	**Comment**
Hardware	HP Superdome	High-end Itanium servers
Disk Storage	5 TB	Available Secondary memory
RAM	40-144 GB	Primary memory available
No. of cores	6-24	Number of cores in the node
No. of Virtual Machines	10-12	For running workloads

6.2 Experimental Results

The proposed ELB model has been evaluated in BSNL data center. The ELB model comprises of the two ELB techniques, one of which is ELB(RU) that considers the

Table 6.4: Software Configuration of each server

Server	System Type	Software
Middleware/Admin/Catalog/CS	Operating System	HP-UX OS
Middleware	Application	Kenan Middleware & BEA Tuxedo
Admin/Catalog/CS	Database	Oracle 9
Admin/Catalog/CS	Application	Kenan FX 1.2

utilization of the resources at its prime level and the other is ELB(w/o RU), whose main focus is on the performance and does not consider the resource utilization directly. Corresponding test cases have been designed for evaluating both these techniques and these cover : the test cases for the resource utilization (both CPU and memory), number of nodes used and migrations performed, the amount of energy consumed, the load balancing and the performance.

In the actual production environment, accounts are distributed on each of the 8 GSM prepaid customer servers in round robin fashion. But in the current experiment, the energy-aware technique is used for deciding the customer server. Note :- The Catalog server is denoted as 1, the Admin server as 2 and the Customer Server number 1 i.e. CS1 as 3. So, Server 17 is CS15, Server 18 is CS16, Server 19 is CS17 and Server 20 is CS18.

Workload description

An account creation in Kenan is not a single transaction. A successful account creation consists of the following jobs :

1. Check for inventory - Once the customer server is selected for an account creation, checks are performed for the inventory.

2. Service creation - After checking the inventory, a service like GSM Prepaid Mobile service, GSM Postpaid Mobile service etc. is created for that account.

3. Package/Plan - Once the service is created for an account, various Packages/Plans (like 525 Package/Plan) are attached to that service.

4. Order generation - An order for all the above services will be generated to be able to deliver them to an account.

5. Workflow generation - After the order generation, workflow is triggered for that type of order. Any workflow consists of different milestones. An order is marked

as complete only if all the milestones and the workflows are completed. Following are the major milestones for creation of a prepaid service under an account :

- MNP Milestones - Checks if the number is MNP number. If yes, it is added to the MNP servers.
- HLR Creation workflow - Number is created on HLR enabling a number for Basic Telephony services e.g Voice.
- SMSC Creation workflow - Number is created on SMSC, for SMS facility.
- SDP Creation workflow - Number is created on SDP, for the facilities like BSNL Live, astrology,jokes etc.
- APN workflow - This adds data services to the number.

Each of the above steps is performed using a number of APIs calls and database transactions, i.e. for performing single step, more than one transactions may be required. Also for an account, data is inserted into multiple tables. Typically 400-500 oracle transactions & API calls are performed for a single account creation on Kenan.

In the existing system, the load balancing is done without considering the utilization of resources and energy consumption, leading to the wastage of resources as well as the energy. To solve this problem, the resource utilization and the energy consumption have been considered in this case study to validate the proposed ELB(RU) technique.

6.3 Experimental Results of ELB(RU) technique

This section evaluates the performance of the proposed ELB(RU) technique in BSNL data center and compares it with the existing non-energy-aware Round Robin (RR) technique and two other standard techniques, FFD & ACO on various parameters, like resource utilization, number of nodes used, number of migrations used, energy consumption and performance.

The ELB(RU) technique considers the resource utilization at the prime level and aims to balance the system load across minimum required active nodes, thereby enhancing resource utilization and hence curtailing the energy consumption of the cloud data centers. The ELB(RU) technique can work from the cloud provider's perspective, as, a cloud provider always wants to utilize the cloud resources efficiently to reduce the cost and the energy consumption of the cloud. Figure 6.3 shows the screenshot to start the actual process of balancing the workloads on various servers.

Figure 6.3: Execution of the Load Balancing Process

The resource utilization is considered in terms of the CPU and the memory utilization as these are the primary sources of energy consumption. As the under-utilization of these resources lead to the energy wastage, therefore, the aim is to improve their utilization in order to prevent the energy wastage and curtail the energy consumption level. The energy consumption is further reduced by trying to reduce the number of nodes. This is done by an attempt to balance the workloads on minimum number of the active nodes, thereby, putting the remaining nodes to sleep mode to save energy. The performance is considered in terms of handling the number of workloads. The more is the number of the workloads handled, the higher is the performance.

6.3.1 Test Case 1: Resource utilization (CPU & Memory)

This subsection deals with the resource utilization of each technique, as the target is to improve the utilization levels of the resources (CPU & memory) to save energy. The percentage of resource utilization for all the four techniques, RR, FFD, ACO and ELB(RU) has been calculated at various instances of time. The percentage utilization of resources in ELB(RU) technique is more as compared to the other three load balancing techniques at different points of time. Figure 6.4 presents the comparative view of the four techniques on the basis of the CPU utilization. The improvement in the CPU utilization level of the servers by ELB(RU) over the rest is clear from the Figure. It shows that ELB(RU) technique is able to enhance the level of CPU utilization by 70.62% over RR, by 52.26% over FFD and by 26.17% over ACO on an average.

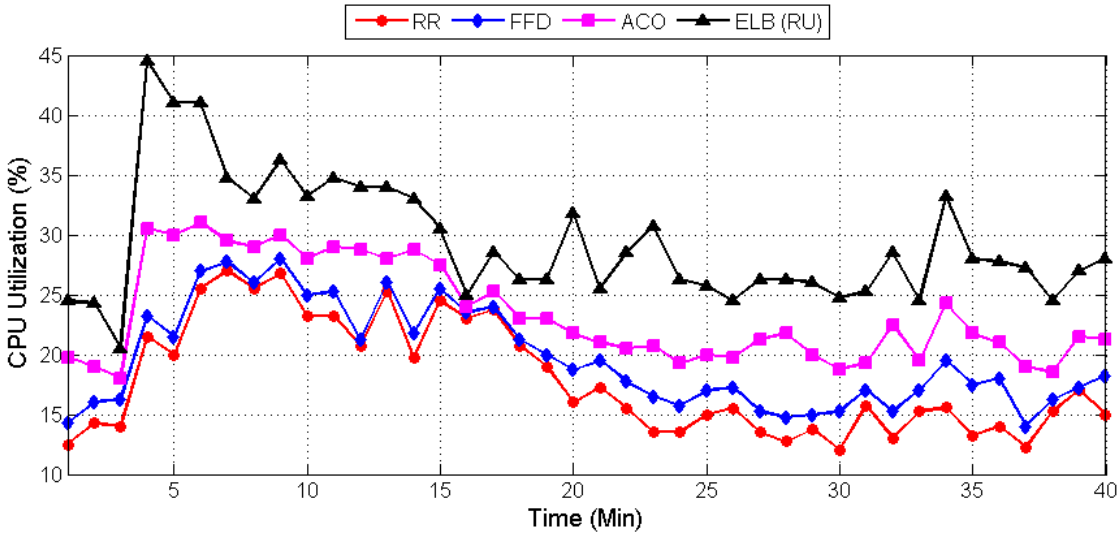
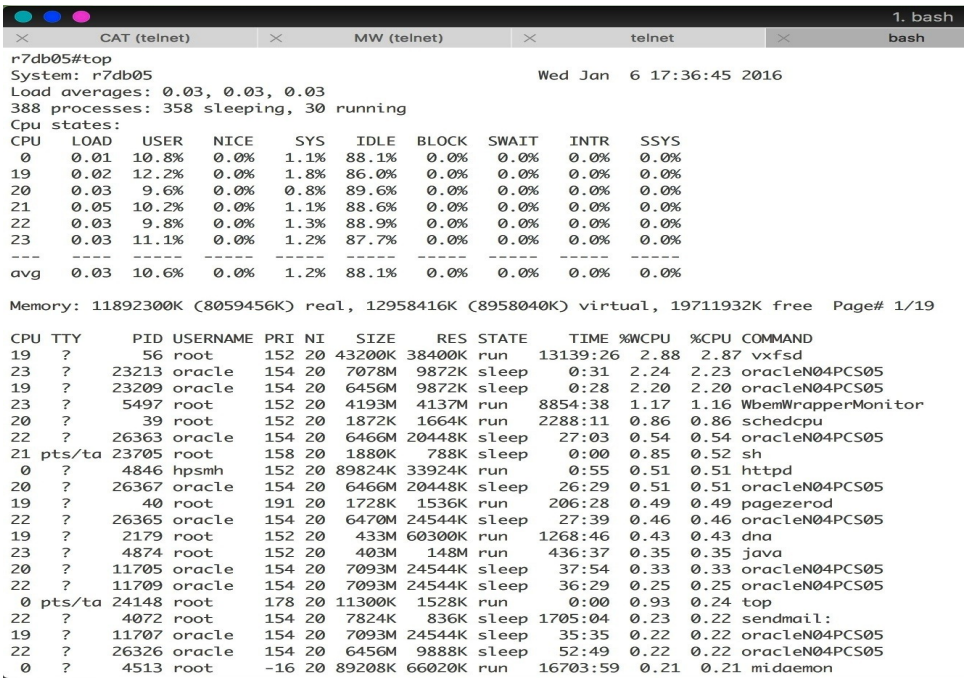


Figure 6.4: Percentage of CPU Utilization by each Technique

Figures 6.5 to 6.8 show the snapshots displaying the CPU utilization of servers by using each of the four techniques. Figure 6.5 is the screenshot to show that about 88% of the CPU is idle when RR technique is run. In other words, only about 12% of CPU is being utilized.



1. bash

CAT (telnet) | MW (telnet) | telnet | bash

```
r7db05#top
System: r7db05                                  Wed Jan  6 17:36:45 2016
Load averages: 0.03, 0.03, 0.03
388 processes: 358 sleeping, 30 running
Cpu states:
CPU    LOAD    USER    NICE     SYS    IDLE   BLOCK   SWAIT    INTR    SSYS
 0     0.01   10.8%    0.0%    1.1%   88.1%    0.0%    0.0%    0.0%    0.0%
19     0.02   12.2%    0.0%    1.8%   86.0%    0.0%    0.0%    0.0%    0.0%
20     0.03    9.6%    0.0%    0.8%   89.6%    0.0%    0.0%    0.0%    0.0%
21     0.05   10.2%    0.0%    1.1%   88.6%    0.0%    0.0%    0.0%    0.0%
22     0.03    9.8%    0.0%    1.3%   88.9%    0.0%    0.0%    0.0%    0.0%
23     0.03   11.1%    0.0%    1.2%   87.7%    0.0%    0.0%    0.0%    0.0%
---    ----   -----   -----   -----   -----   -----   -----   -----   -----
avg    0.03   10.6%    0.0%    1.2%   88.1%    0.0%    0.0%    0.0%    0.0%

Memory: 11892300K (8059456K) real, 12958416K (8958040K) virtual, 19711932K free  Page# 1/19

CPU TTY      PID USERNAME PRI NI    SIZE     RES STATE     TIME %WCPU  %CPU COMMAND
19   ?        56 root     152 20 43200K 38400K run   13139:26  2.88  2.87 vxfsd
23   ?     23213 oracle   154 20  7078M  9872K sleep     0:31  2.24  2.23 oracleN04PCS05
19   ?     23209 oracle   154 20  6456M  9872K sleep     0:28  2.20  2.20 oracleN04PCS05
23   ?      5497 root     152 20  4193M  4137M run    8854:38  1.17  1.16 WbemWrapperMonitor
20   ?        39 root     152 20  1872K  1664K run    2288:11  0.86  0.86 schedcpu
22   ?     26363 oracle   154 20  6466M 20448K sleep    27:03  0.54  0.54 oracleN04PCS05
21 pts/ta 23705 root     158 20  1880K   788K sleep     0:00  0.85  0.52 sh
 0   ?      4846 hpsmh    152 20 89824K 33924K run       0:55  0.51  0.51 httpd
20   ?     26367 oracle   154 20  6466M 20448K sleep    26:29  0.51  0.51 oracleN04PCS05
19   ?        40 root     191 20  1728K  1536K run     206:28  0.49  0.49 pagezerod
22   ?     26365 oracle   154 20  6470M 24544K sleep    27:39  0.46  0.46 oracleN04PCS05
19   ?      2179 root     152 20   433M 60300K run    1268:46  0.43  0.43 dna
23   ?      4874 root     152 20   403M   148M run     436:37  0.35  0.35 java
20   ?     11705 oracle   154 20  7093M 24544K sleep    37:54  0.33  0.33 oracleN04PCS05
22   ?     11709 oracle   154 20  7093M 24544K sleep    36:29  0.25  0.25 oracleN04PCS05
 0 pts/ta 24148 root     178 20 11300K  1528K run       0:00  0.93  0.24 top
22   ?      4072 root     154 20  7824K   836K sleep 1705:04  0.23  0.22 sendmail:
19   ?     11707 oracle   154 20  7093M 24544K sleep    35:35  0.22  0.22 oracleN04PCS05
22   ?     26326 oracle   154 20  6456M  9888K sleep    52:49  0.22  0.22 oracleN04PCS05
 0   ?      4513 root     -16 20 89208K 66020K run    16703:59  0.21  0.21 midaemon
```

Figure 6.5: CPU Utilization of RR

Figure 6.6 illustrates that about 28% of the CPU is being utilized when the FFD technique is run, keeping the CPU idle by about 72%.

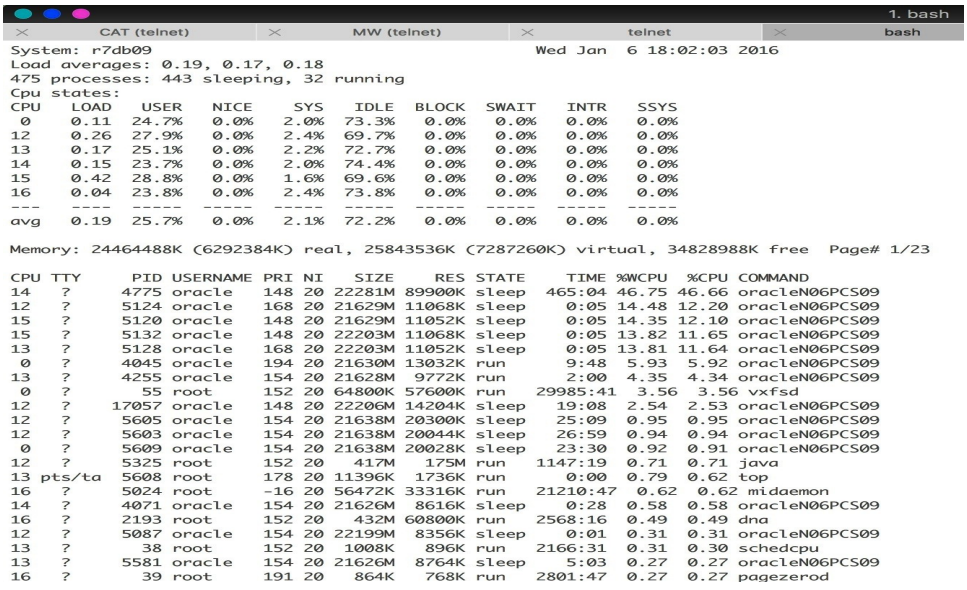


Figure 6.6: CPU Utilization of FFD

Figure 6.7 informs that about 69% of the CPU remains idle whereas only about 31% of the CPU is being used when ACO is run.

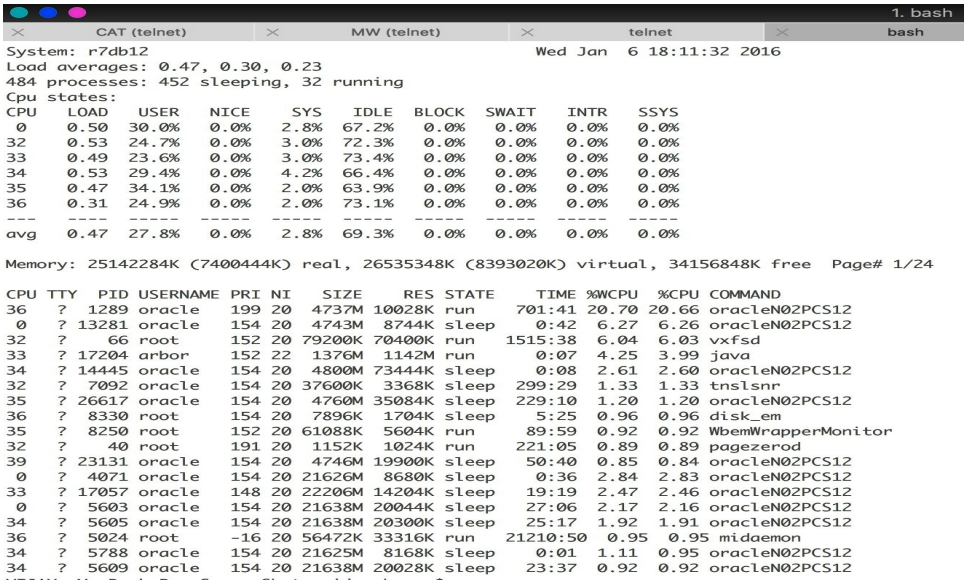


Figure 6.7: CPU Utilization of ACO

Figure 6.8 indicates that when ELB(RU) is run, about 45.5% of the CPU is being utilized leaving about 55.5% of the CPU idle. All these screen shots depict that CPU utilization is improved from 12% in RR to 28% in FFD, then to 31% in ACO but to the maximum 45.5% in ELB(RU).

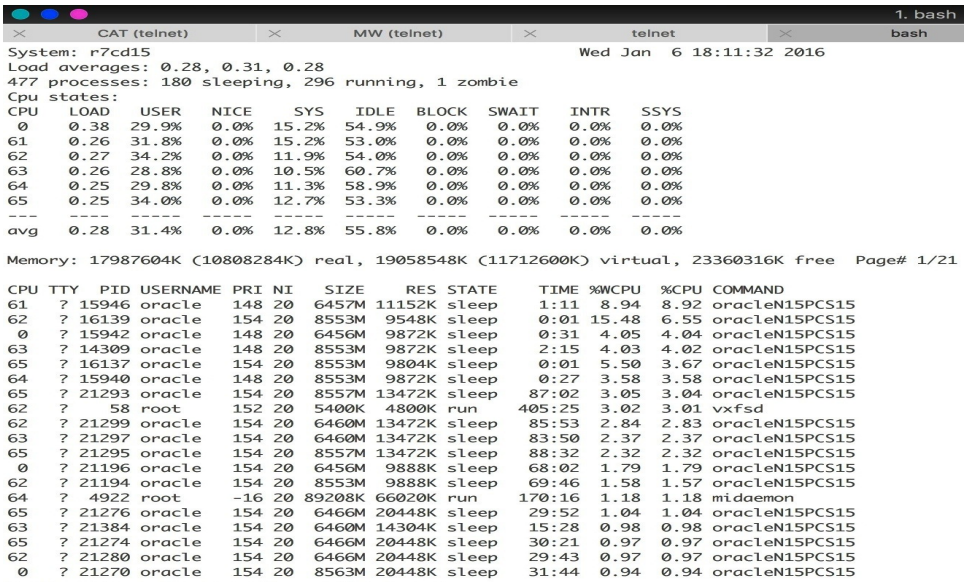


Figure 6.8: CPU Utilization of ELB(RU)

Figure 6.9 interprets the betterment in the memory utilization level of servers by using ELB(RU) over RR, FFD & ACO by 32.86%, 26.76% & 13.63% respectively.

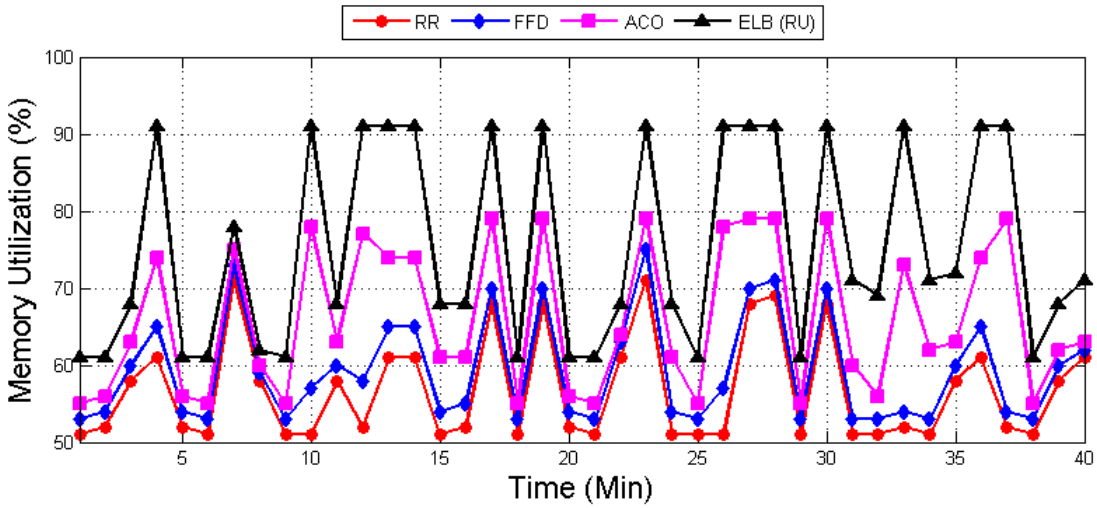


Figure 6.9: Percentage of Memory Utilization by each Technique

Figures 6.10 to 6.13 exhibit the memory utilization of servers by using all the four approaches. Figure 6.10 guides that running the RR technique keeps about 39% of the memory idle.

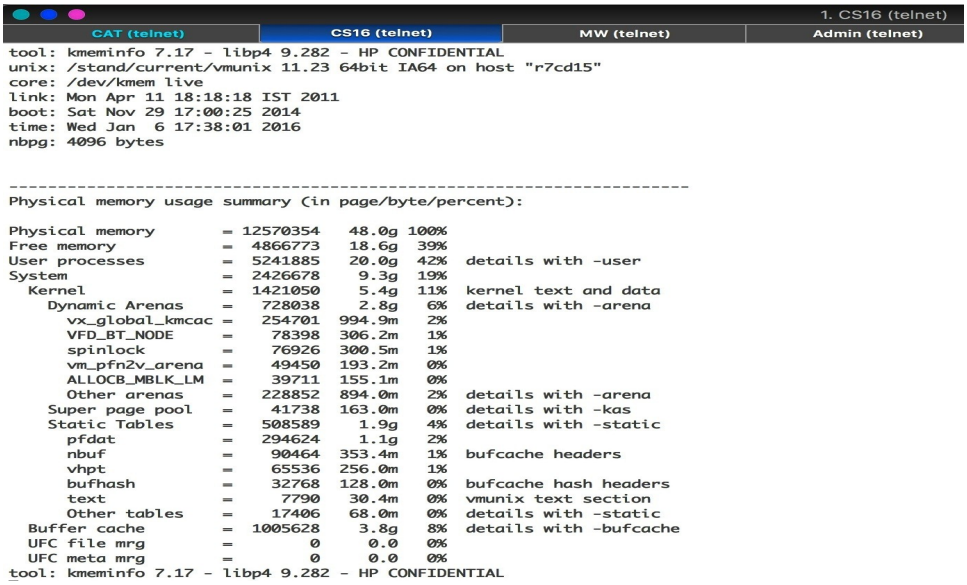


Figure 6.10: Memory Utilization of RR

Whereas the idleness of the memory reduces to 25% when FFD is run as shown in Figure 6.11.

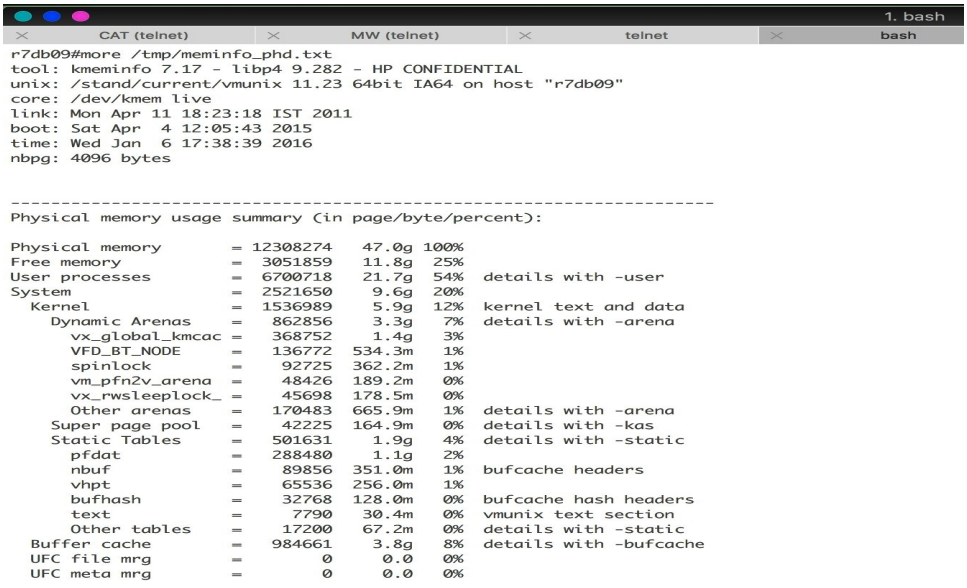


Figure 6.11: Memory Utilization of FFD

Figure 6.12 explains that about 20% of the memory sits idle when ACO is run. Whereas it reduces to only 10% when using the ELB(RU) technique as displayed in

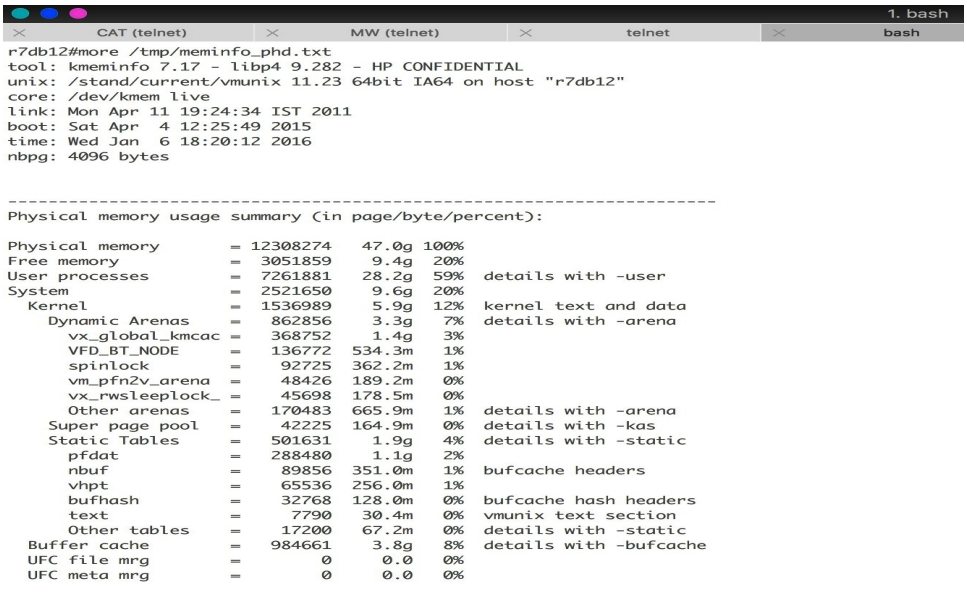


Figure 6.12: Memory Utilization of ACO

Figure 6.13. All the snapshots prove that the memory utilization level has been improved from 61% in RR to 75% in FFD, then to 80% in ACO and the maximum to 90% in ELB.

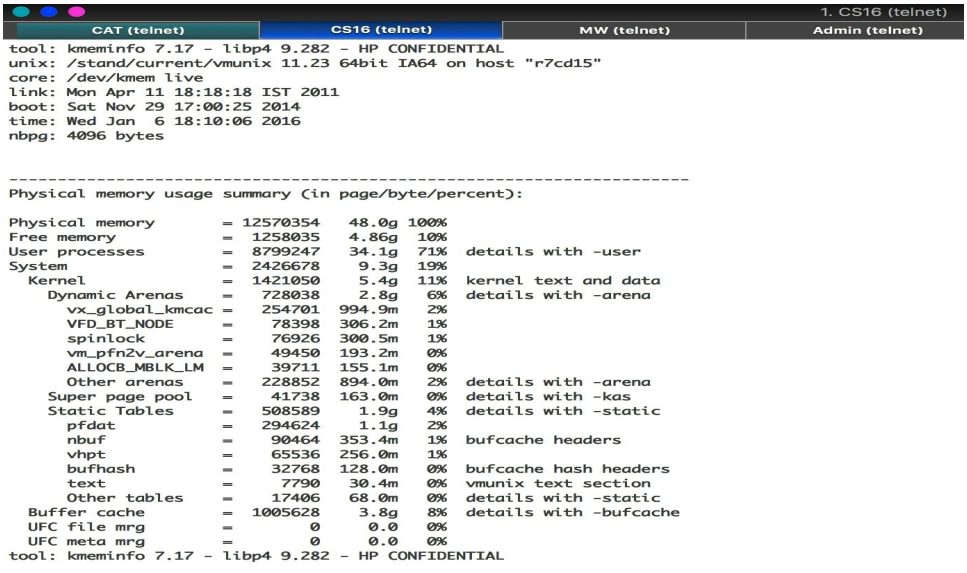


Figure 6.13: Memory Utilization of ELB(RU)

The improvement in the CPU & memory utilization levels of servers by ELB(RU) is due to the fact that it makes use of the ERU model (discussed in Chapter 3) to make the best use of the resources. When ELB(RU) is run, the workload is processed on that server which satisfies the conditions imposed by ERU. Whereas, when RR is run, the workloads are processed in a round robin fashion without checking the utilization levels of resources. This technique keeps all the nodes in an on state with usually low utilization, leading to the wastage of resources. In the case of FFD, the workloads are first sorted in a decreasing order and are then processed by those servers which can process them but without a check on the resource utilization of the servers, leading to their wastage. When ACO is run, it processes the workloads using its probabilistic nature thereby utilizing the resources better than RR and FFD but less than the proposed ELB(RU) technique.

6.3.2 Test Case 2: VM Migrations

This subsection considers VM migrations followed by each of the four techniques, RR, FFD, ACO & ELB(RU), in order to reduce the number of used nodes. The saved nodes can then be put to sleep mode to save energy. Figure 6.14 points out the number of VM migrations performed by each technique. RR technique does not use any kind of VM

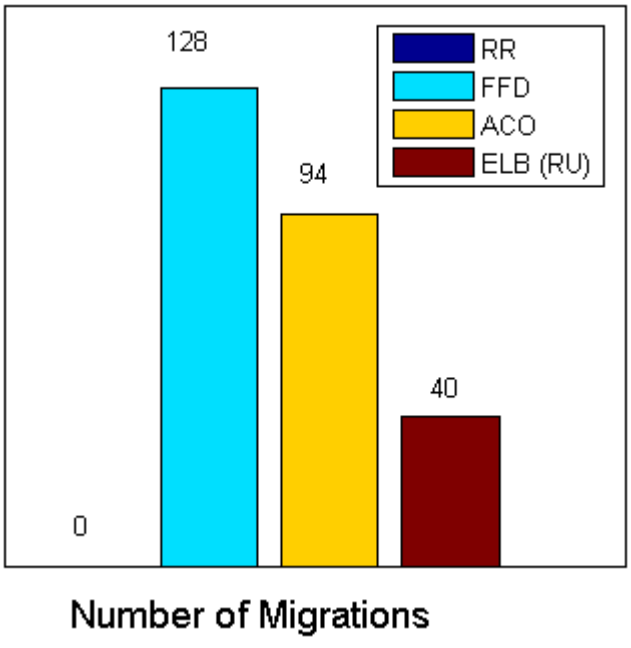


Figure 6.14: VM Migrations Performed by each Technique

migrations to balance the load. It simply creates all the accounts, on all the servers equally, in a round-robin fashion. Therefore, the number of the VM migrations in the case of RR approach has been shown as nil. The other three approaches, in trying to balance the load on lesser number of active nodes, perform a number of VM migrations which varies in all the three techniques. The maximum number of VM migrations is carried out in FFD approach, which further reduces in ACO approach and the minimum number of VM migrations is executed in ELB(RU) approach. ELB(RU) technique is able to reduce 68.75% of VM migrations over FFD and 57.45% of VM migrations over ACO on an average. This is because ELB(RU) approach uses ABC-based ERU to achieve optimal resource utilization, and then uses FFO-EVMM technique to migrate VMs in an energy-aware manner from optimally utilized resources only, leading to a very few migrations. The betterment in the resource utility levels curtails the number of VM migrations thereby averting the wastage of the energy.

6.3.3 Test Case 3: Nodes Used

It is eminent to monitor the number of active hosts in the system to avoid the situations of the idle hosts consuming unnecessary power, thereby, violating the least energy requirement criterion. This subsection informs about the nodes used by each technique

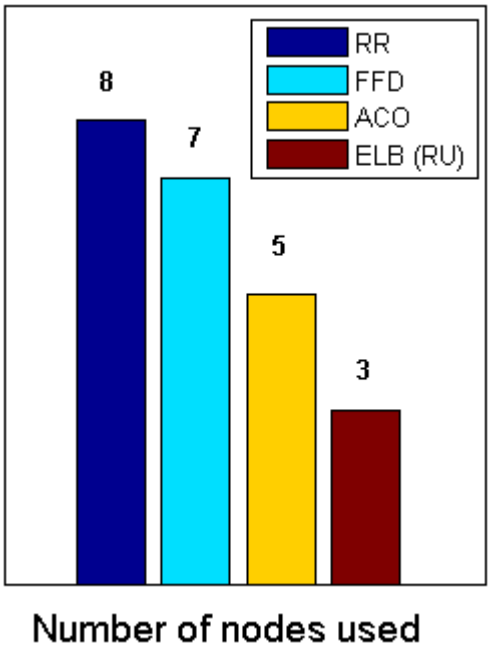


Figure 6.15: Nodes Used by each Technique

to balance the load of the system. Figure 6.15 provides a comparative view of the number of nodes used by RR, FFD, ACO and ELB(RU). The lesser the number of nodes used, the lesser is the energy consumption as the rest of nodes will be in sleep mode state to save energy. It is surveyed that if a node is on and is less utilized, it consumes more energy than when it is fully utilized. So, it is always better to shift the load of an under-utilized node to some other active node and keep it in sleep mode to save energy. It is clear from Figure 6.15 that RR approach uses the maximum number of nodes as compared to the other three approaches. This is because, the RR approach does not consider the utilization of the resources of the nodes and creates the accounts on all the nodes equally. Then is FFD technique that uses an average of 12.5% less nodes than RR technique but uses 28.57% more nodes than the ACO and 57.14% more nodes than the ELB(RU). After that is ACO technique which uses 37.5% and 28.57% less nodes than the nodes used by RR & FFD approaches respectively, but uses 40% more nodes than ELB(RU) technique. ELB(RU) approach uses the least number of the nodes due to the use of ERU. ELB(RU) achieves an average saving of about 62.50% nodes when compared to RR, about 57.14% nodes when compared to FFD and about 40% nodes when compared to ACO. Also, the ELB(RU) uses an average of 53.21% less nodes than the other three techniques.

6.3.4 Test Case 4: Energy Consumption

As the main purpose of this work is to reduce the energy consumption of a data center by using an energy-aware solution, therefore, this subsection discusses about the energy consumed by all the four techniques, RR, FFD, ACO and ELB(RU). Figure 6.16 portrays that the energy consumption of the data center by using ELB(RU) technique is much less when compared to the other three techniques. The energy consumption in RR approach is the maximum and remains almost the same throughout as this approach uses maximum number of servers without considering their utilization, thereby consuming more energy. In the case of FFD, the consumed energy is 9.06% lesser than RR approach but 14.11% more than ACO and 41.84% more than ELB(RU). This is because FFD uses less number of nodes than used by RR technique but more than ACO and ELB(RU). Also, the number of VM migrations used by FFD approach is more than ACO and ELB(RU). ACO approach consumes 21.90% & 14.11% of less energy as compared to RR & FFD respectively as it uses less nodes than both the techniques hence saving energy. Also, it saves energy by following less VM migrations than FFD. It further makes use of more nodes & follows more VM migrations than ELB(RU) ap-

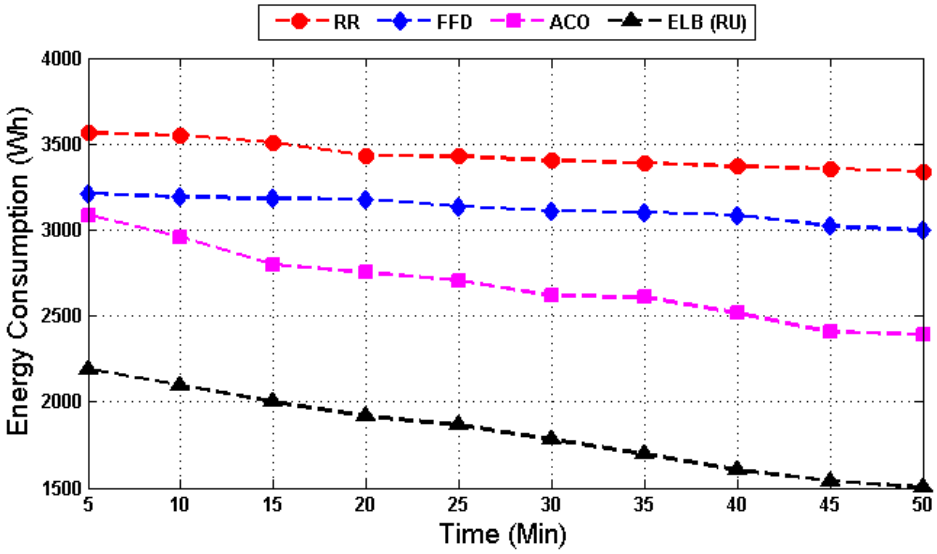


Figure 6.16: Energy Consumed by each Technique

proach, therefore consumes 32.45% more energy than ELB(RU) technique. ELB(RU) approach uses the least number of nodes than the rest of the three techniques and executes the lowest number of migrations than FFD & ACO, therefore, it consumes the minimum energy. It saves an average of 47.12% , 41.84% & 32.45% of energy over RR, FFD & ACO respectively. Also, an overall average energy saving of 40.47% is achieved by ELB(RU) technique. Figures 6.17 to 6.20 exhibit the decrease in energy consumption level through snapshots.

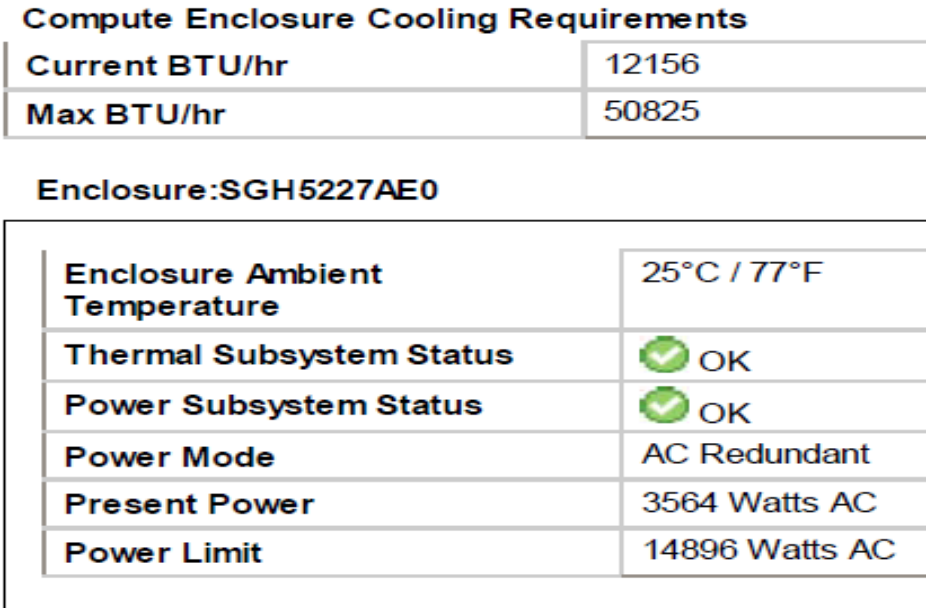


Figure 6.17: Energy Consumption of RR

Compute Enclosure Cooling Requirements

Current BTU/hr	10955
Max BTU/hr	50825

Enclosure:SGH5227AE0

Enclosure Ambient Temperature	25°C / 77°F
Thermal Subsystem Status	OK
Power Subsystem Status	OK
Power Mode	AC Redundant
Present Power	3212 Watts AC
Power Limit	14896 Watts AC

Figure 6.18: Energy Consumption of FFD

Compute Enclosure Cooling Requirements

Current BTU/hr	10532
Max BTU/hr	50825

Enclosure:SGH5227AE0

Enclosure Ambient Temperature	26°C / 78.8°F
Thermal Subsystem Status	OK
Power Subsystem Status	OK
Power Mode	AC Redundant
Present Power	3087 Watts AC
Power Limit	14896 Watts AC

Figure 6.19: Energy Consumption of ACO

Compute Enclosure Cooling Requirements	
Current BTU/hr	7472
Max BTU/hr	50825

Enclosure:SGH5227AE0	
Enclosure Ambient Temperature	36°C / 96.8°F
Thermal Subsystem Status	OK
Power Subsystem Status	OK
Power Mode	AC Redundant
Present Power	2190 Watts AC
Power Limit	14896 Watts AC

Figure 6.20: Energy Consumption of ELB(RU)

6.3.5 Test Case 5: Load Balancing

The main process of balancing the load of the system, by all the four techniques, has been presented in this subsection. It is shown that when RR technique is run, every time a new workload is processed by the next node in the order without checking the resource utilization or energy consumption of the servers. Though, by this technique, the load is balanced effectively across the nodes, but nodes are left in an on state with usually low utilization, leading to the resource and the energy wastage. Also, this technique uses the maximum number of servers. The FFD approach sorts the workloads in a decreasing order before processing them on those servers which can handle them. This helps the FFD technique to balance the load on a fewer servers than the RR approach but more than ACO & ELB(RU). It also does not check resource and energy utilization of the servers, leading to their wastage. In the case of ACO, the workloads are processed using the probabilistic nature of the algorithm. Hence, it behaves slightly better by balancing the load of the system on lesser number of servers than RR & FFD but more than ELB(RU). Hence, it utilizes the resources and saves the energy of the data center better than RR & FFD, but less than the proposed technique, ELB(RU). Whereas, in the proposed ELB(RU) technique, firstly, ERU is used to optimally utilize the resources by allocating the workloads to the most energy-aware nodes. Later, if at the run-time, when the energy consumption of the nodes

start increasing beyond a threshold value, it makes use of its FFO-EVMM migration technique for performing energy-aware VM migrations to avoid hot spots and to balance the load on the minimum number of active servers, thereby, switching the rest of the servers to sleep mode, hence lowering the energy consumption levels. Figure 6.21 shows

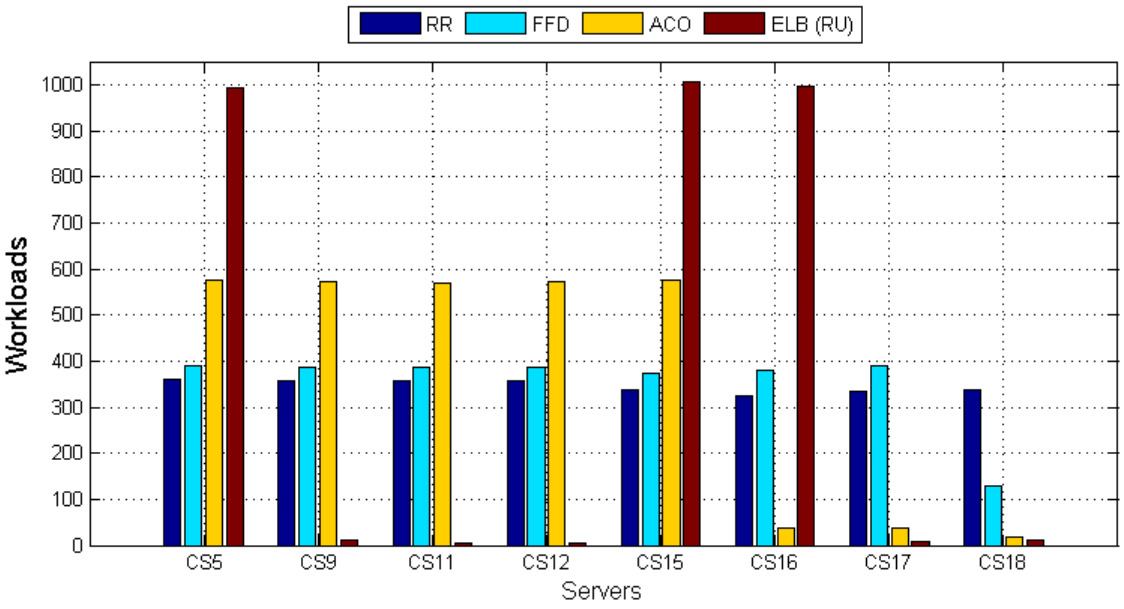


Figure 6.21: Servers Used by each Technique before Migration

the workloads processed and the servers used by each technique. This figure further depicts that the existing RR approach uses the more number of the servers as it is processing the workloads in a round robin fashion. Whereas the proposed ELB(RU) approach uses the less number of the servers as the workloads are processed only after checking the utilized resources and the consumed energy. The less number of used nodes, helps in reducing the energy consumption of the data center.

Figure 6.22 depicts the load balancing done by each technique using the VM migrations. As the VM migration also consumes energy, so lower is the number of the VM migrations, the lesser is the energy consumed. The figure portrays that RR balances the workload on all the servers equally. Also, it does not perform any VM migration, but still it consumes the maximum energy. This is because in RR approach, the servers are under-utilized and are in the "on" state, consuming a lot of energy. FFD technique balances the workload on fewer servers than RR but more than ACO and ELB(RU) approaches. The Figure also points out that FFD uses the maximum number of VM migrations. Then is ACO technique which balances the workload even on lesser number of servers. These servers are less than the servers used by RR & FFD approaches but are more than the servers used by ELB(RU) technique. It is clear from the Figure that

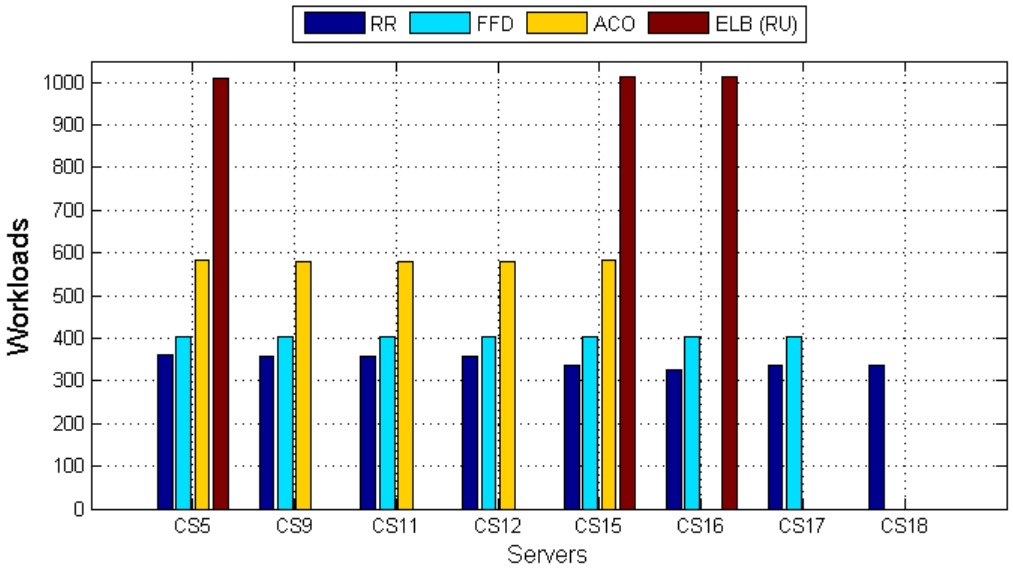


Figure 6.22: Servers Used by each Technique after Migration

the number of VM migrations performed by ACO is less than FFD technique but more than ELB(RU) technique. Last is the load balancing done by the proposed ELB(RU) approach which uses the minimum number of the active servers to balance the load as it is optimally utilizing all the resources. Also, it uses the minimum number of the VM migrations as the VMs are migrated from the optimally utilized resources only. Figures 6.23 & 6.24 show the snapshots for the accounts created by RR approach. It is

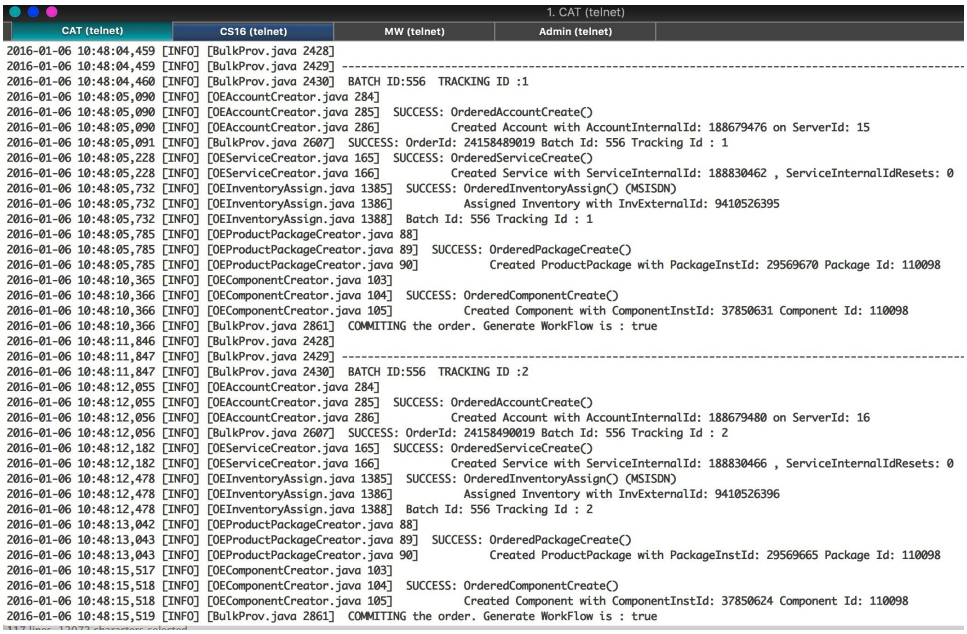


Figure 6.23: Servers Used by RR

clear from the snapshots that every time the new account is created on the new server. e.g. the first account is created first on server id 17, the next on 18, the next on 19 and then on 20, i.e. in a round robin fashion.

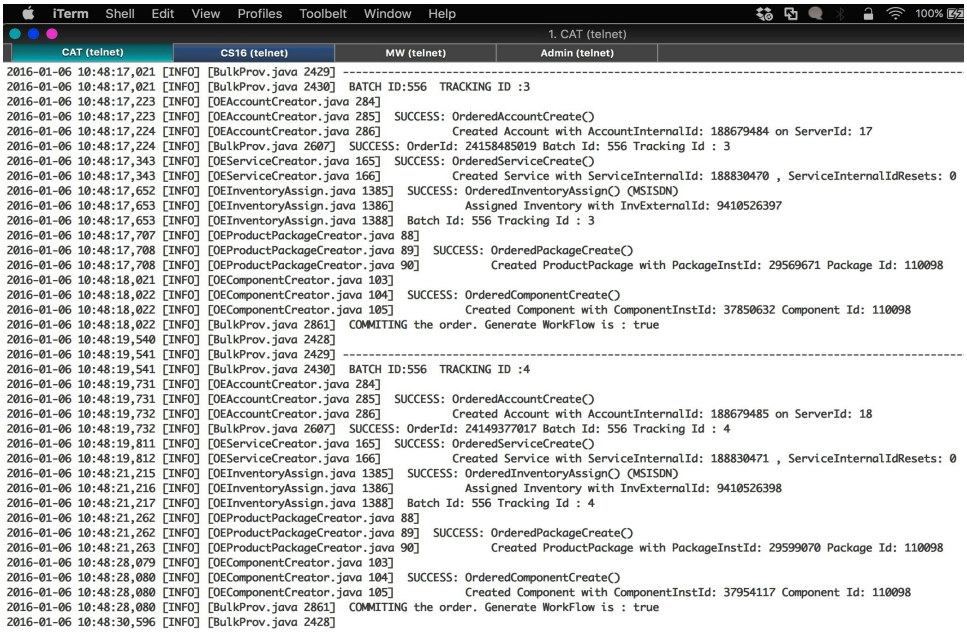


Figure 6.24: Servers Used by RR

Figures 6.25 & 6.26 give the snapshots for the account creation by the FFD and ACO approaches respectively.

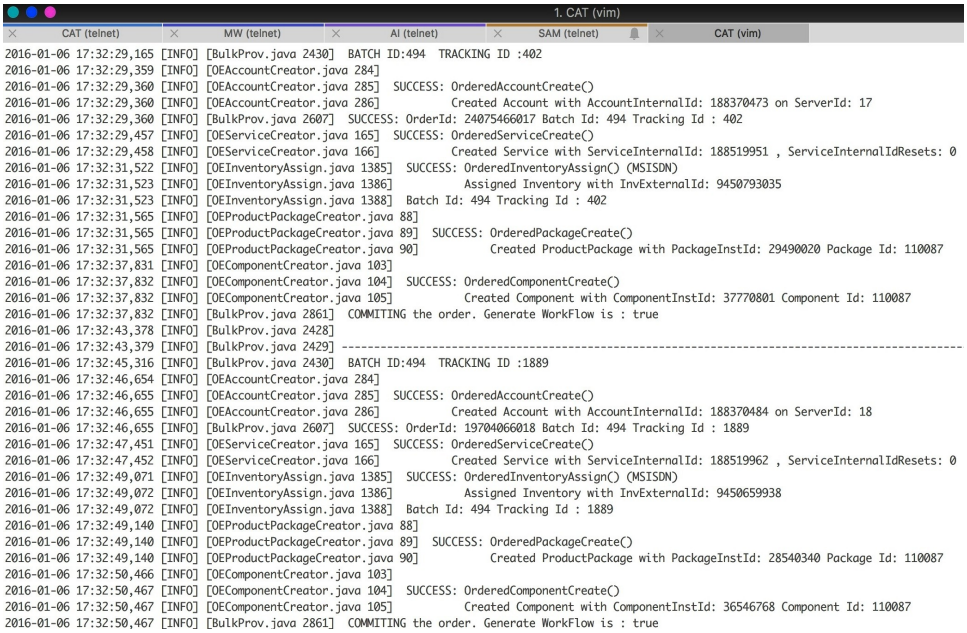


Figure 6.25: Servers Used by FFD

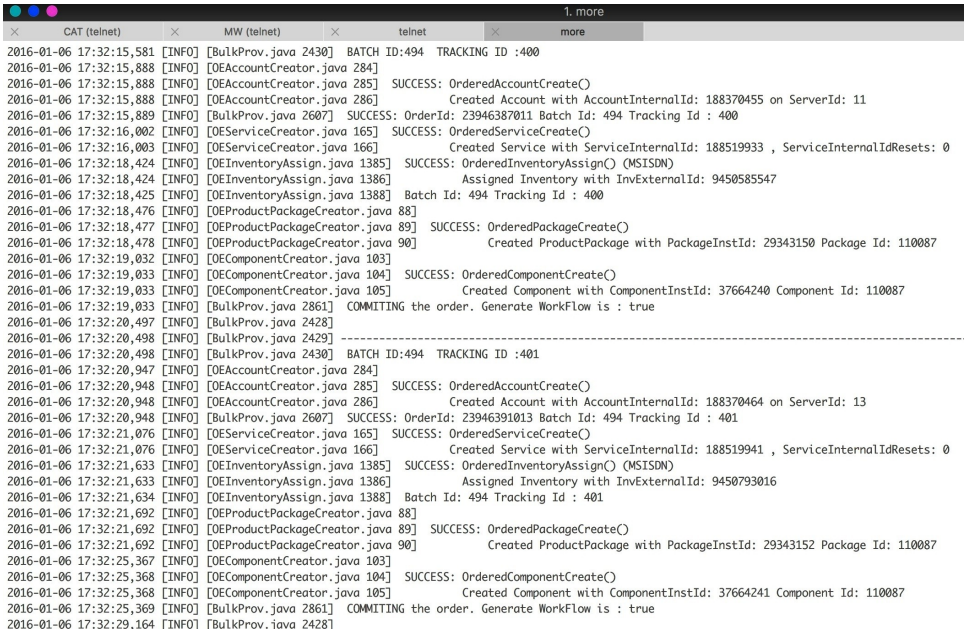


Figure 6.26: Servers Used by ACO

The accounts created by the ELB(RU) technique have been presented in Figures 6.27 & 6.28, where it is clearly seen that the new account is not created on the next server every time, rather it is created on the server only after checking the specific requirements. e.g. the new account is created on server id 17, the next again on 17, after checking the resource utilization and the energy consumption of the servers.

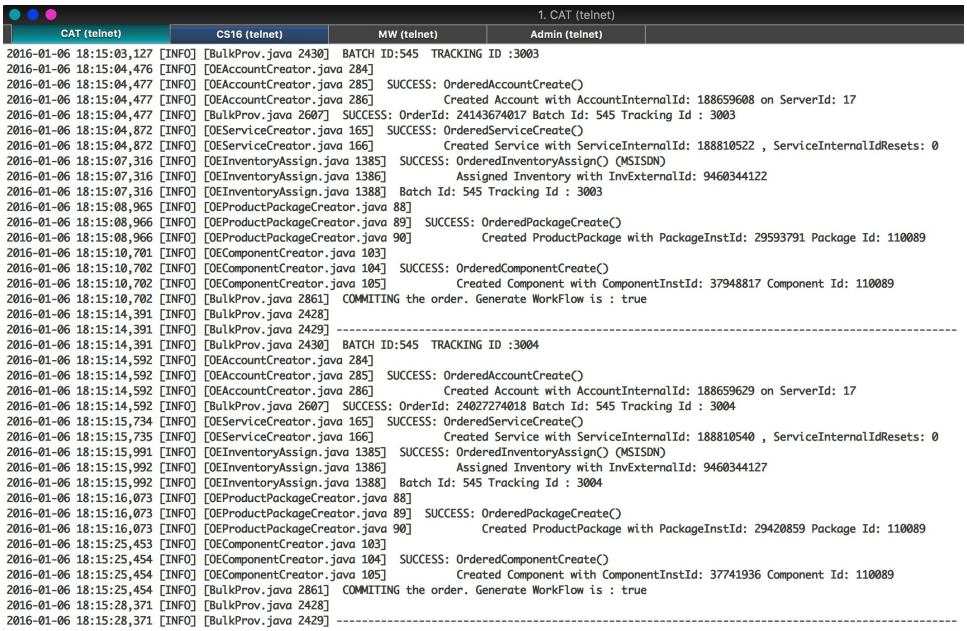


Figure 6.27: Servers Used by ELB(RU)

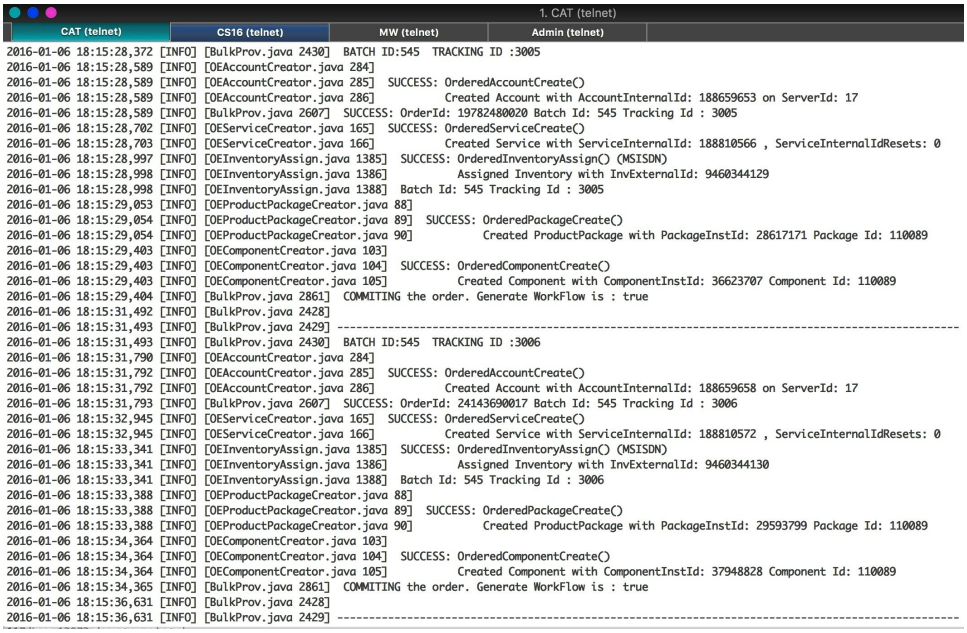


Figure 6.28: Servers Used by ELB(RU)

6.3.6 Test Case 6: Performance

As the performance is considered in terms of handling the number of workloads, Figure 6.29, shows the number of workloads handled by each technique, i.e. the performance of each approach. The Figure clearly shows that the proposed ELB(RU) technique, has handled more workloads than the other three approaches. In other words, the performance of ELB(RU) is the best. ELB(RU) handles an average of about 10.25% more workloads than RR, 7.98% than FFD and 4.78% than ACO. Although,

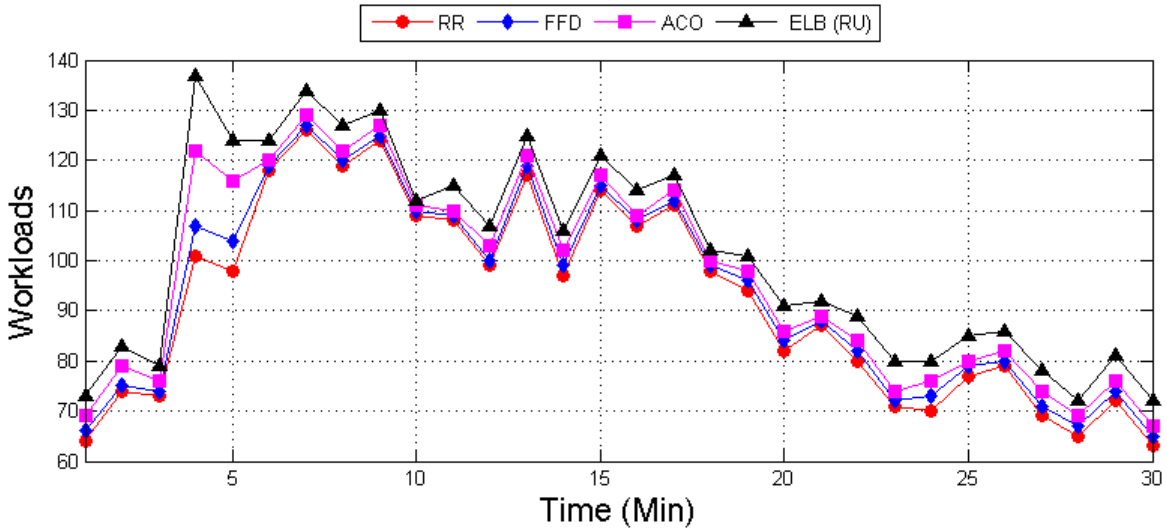


Figure 6.29: Performance of each Technique in terms of Handled Workloads

an overall enhancement in ELB(RU)'s performance is only 7.67%, but this increase is also satisfactory when compared to the saved resources, nodes and the energy.

6.3.7 Discussion

This subsection discusses the improvement of all the three techniques i.e. ELB(RU), FFD and ACO over RR technique, betterment of ELB(RU) approach over the other three individually and the overall enhancement of the proposed ELB(RU) algorithm. Figure 6.30 depicts that there is an improvement of 70.62% in the CPU utilization and 32.86% in the memory utilization followed by 62.5% of saving in nodes on an average.

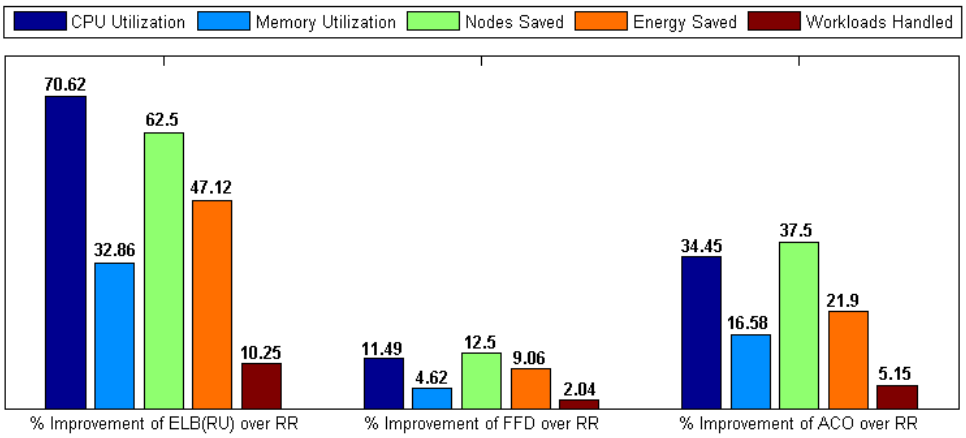


Figure 6.30: Percentage Improvement of FFD, ACO & ELB(RU) over RR

All the three factors lead to 47.12% average reduction in energy consumption. It also shows that an average of 10.25% more workloads are handled when ELB(RU) approach is compared against RR. This figure further portrays the improvement of FFD over RR by showing 11.49% & 4.62% average improvement in CPU & memory utilization respectively, 12.5% & 9.06% of nodes & energy savings respectively and 2.04% increase in handled workloads. The comparison of ACO with RR shows that ACO handles an average of 5.15% more workloads. The CPU utilization is enhanced by 34.45% and the memory utilization is improved by 16.58%. The ACO is further able to save 37.5% of nodes thereby saving 21.90% of energy on an average.

Figure 6.31 outlines the betterment of the ELB(RU) approach over the other three approaches, individually. The comparison of ELB(RU) with RR is the same as stated above and depicted by Figure 6.30. ELB(RU)'s comparison with FFD shows its improvement by figuring out an average increase in CPU & memory utilization of 52.26% & 26.76% respectively. There is an average reduction of 68.75%, 57.14% & 41.84%

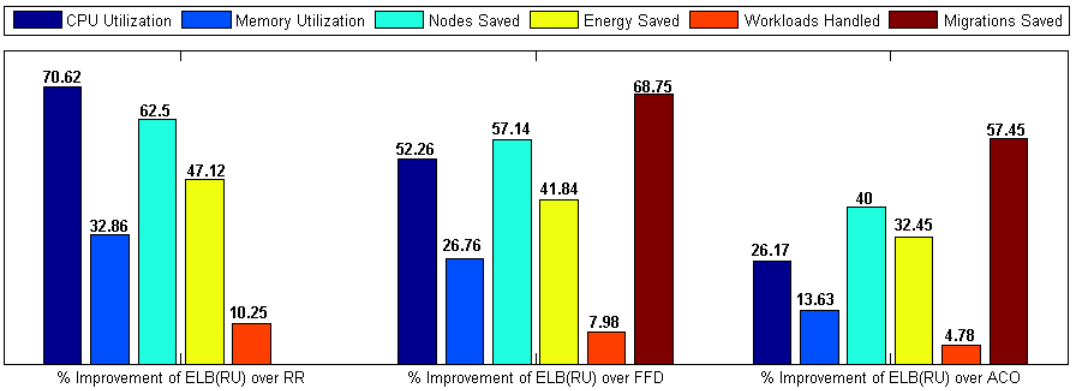


Figure 6.31: Percentage Improvement of ELB(RU) over RR, FFD & ACO

in VM migrations, node & energy usage respectively. Also, ELB(RU) handles 7.98% more workloads than FFD on an average. When ELB(RU) is compared against ACO, it enhances the CPU and memory utilization levels by 26.17% & 13.63% respectively. It further curtails the number of VM migrations, number of used nodes and consumed energy by 57.45%, 40% & 32.45% respectively. There is also an increment of 4.78% in the handled workloads on an average.

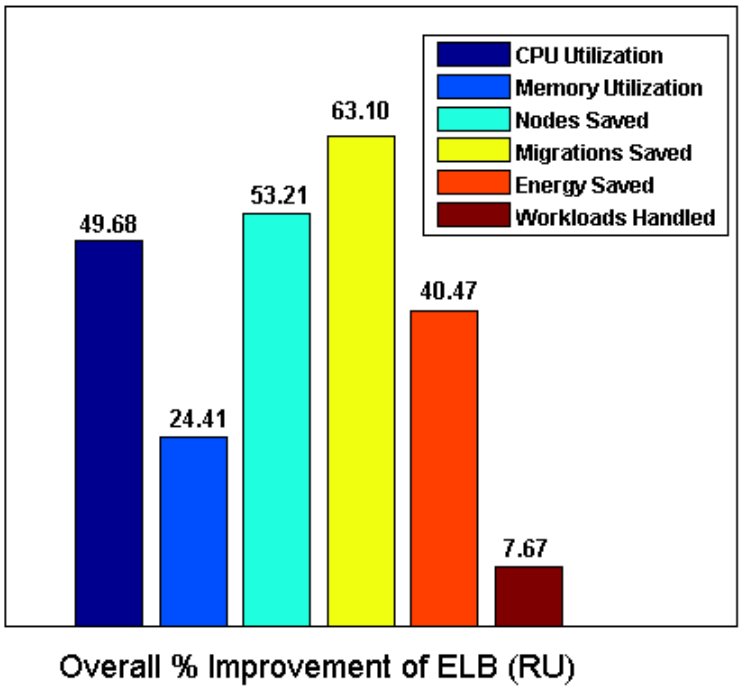


Figure 6.32: Overall Enhancement Graph of ELB(RU)

Figure 6.32 sketches the overall enhancement of the proposed ELB(RU) algorithm. When the performance of the ELB(RU) algorithm is compared against the rest of the three algorithms, it illustrates the overall average enhancement of 49.68% in the CPU utilization level and 24.41% in the memory utilization level. It further cuts down the number of VM migrations by 63.10% and number of node usage by 53.21% on an average. All these factors together help it to curtail the average energy consumption level by 40.47%. Also, an overall improvement in handled workloads is 7.67%.

Hence, it is clear from the above stated results that the proposed ELB(RU) approach is better than the three RR, FFD, ACO techniques in improving the CPU & memory utilization levels, in reducing the number of VM migrations & nodes thereby dropping the energy consumption levels and making data center more energy-aware. The performance of ELB(RU) in terms of handled workloads is also better than the rest.

This section depicted the performance analysis of the proposed ELB(RU)), in BSNL data center followed by a detailed discussion on the competence of the proposed technique over the existing RR, FFD & ACO approaches. The next section draws the details of the verification and the implementation of the proposed ELB(w/o RU) followed by a detailed discussion of its comparison with RR, FFD & ACO and ELB(RU).

6.4 Experimental Results of ELB(w/o RU) Technique

This section evaluates the proposed ELB(w/o RU) technique and compares it with the four techniques namely ELB(RU), RR, FFD & ACO in BSNL data center. Unlike the ELB(RU) technique, the prime focus of the ELB(w/o RU) technique is to achieve a higher user satisfaction level, which can be obtained by handling the more number of the workloads, received from the cloud users. Although, both the ELB techniques aim to balance the system load across minimum required active nodes in the cloud data center, thereby enhancing resource utilization and performance levels and hence curtailing the energy consumption of the cloud data centers. The ELB(w/o RU) technique can work from the cloud user's perspective, as, a cloud user always wants the system to meet the performance levels. The performance of the proposed ELB(w/o RU) technique has been evaluated and compared with the four techniques on various parameters, like performance, energy consumption, number of nodes used and VM migrations performed.

6.4.1 Test Case 1: Performance

As the main criterion of this technique is to obtain a high performance level that can be achieved by handling more number of workloads. Figure 6.33, shows the number of workloads handled by each technique, i.e. the performance of each approach. The figure clearly depicts that the proposed ELB(w/o RU) technique has handled the maximum number of workloads, satisfying the more cloud users and making the best performance.

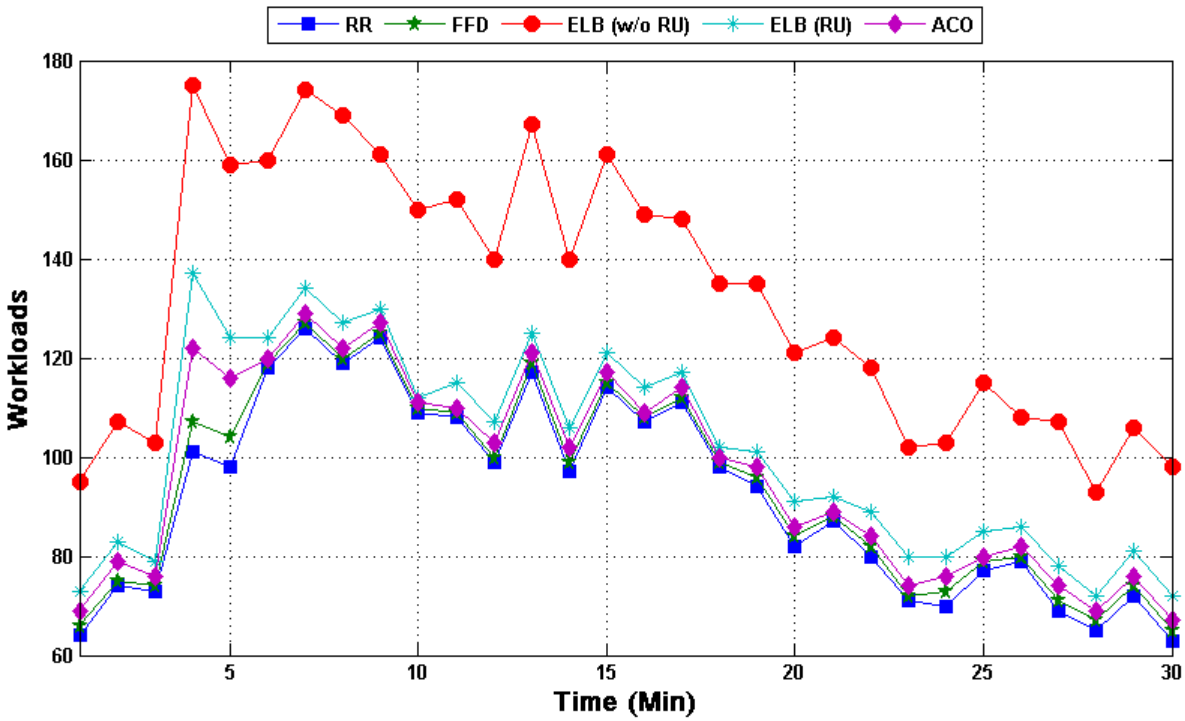


Figure 6.33: Number of Workloads Handled by each Technique

On an average, it has dealt with 44.42% more workloads than RR, 41.46% than FFD, 37.28% than ACO and 31.03% more workloads than ELB(RU). The reason to achieve this is that ELB(w/o RU) assigns the workload to the first available node in the system fulfilling its resource requirements, helping in enhancing the performance as workloads are executed on the resources fulfilling their requirements, thereby saving time in allocating the workloads to the nodes. This further reduces the waiting time of the workloads, hence aiding ELB(w/o RU) in executing the workloads faster and handling more workloads. So, the more the number of workloads handled, the higher is the user satisfaction level and so is the performance.

6.4.2 Test Case 2: VM Migrations

The VM migrations are performed to migrate the VMs from one active node to the other active node, when the energy consumption of that node exceeds a particular threshold. The node from which the VMs are migrated can be put in sleep mode to save energy. This subsection considers VM migrations followed by each of the five techniques, RR, FFD, ACO, ELB(RU) & ELB(w/o RU), thereby switching the saved nodes to sleep mode, hence reducing energy consumption. Figure 6.34 points out the number of VM migrations performed by each technique.

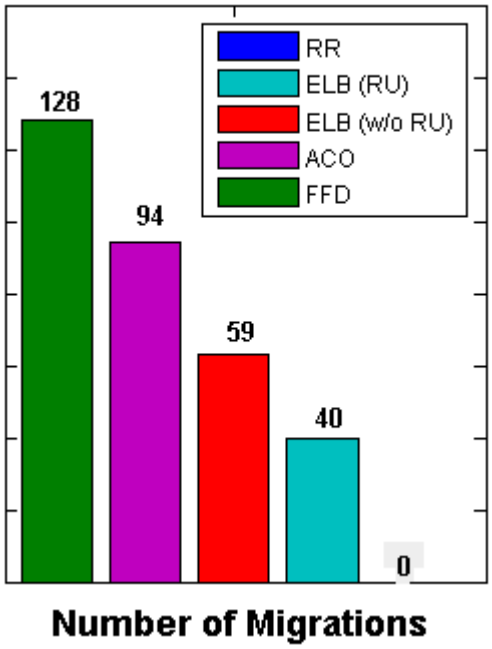


Figure 6.34: Number of VM Migrations Performed by each Technique

The proposed ELB(w/o RU) technique performs 53.91% less migrations than FFD and 37.23% less than ACO but 32.20% more than ELB(RU). The reason for ELB(w/o RU) to reduce the VM migrations over FFD & ACO is that it uses FFO-EVMM migration technique which optimally migrates the VMs, thereby reducing the number of VM migrations. The reduced number of VM migrations helps in reducing energy consumption too. However, ELB(w/o RU) carries out 32.20% more VM migrations than ELB(RU), this is because ELB(RU) approach uses ERU to achieve optimal resource utilization, and then uses FFO-EVMM technique to migrate the VMs from optimally utilized resources only, leading to a very few migrations. The enhancement in the resource utility levels lowers the number of VM migrations thereby curtailing the wastage of energy.

6.4.3 Test Case 3: Nodes Used

As it is very important to keep a track of the active hosts to prevent the conditions of overloading and under-loading, thereby preventing energy wastage, this subsection enlightens the nodes used by each technique to balance the load of the system . Figure 6.35 provides a comparative view of the number of nodes used by RR, FFD, ACO, ELB(RU) and ELB(w/o RU).

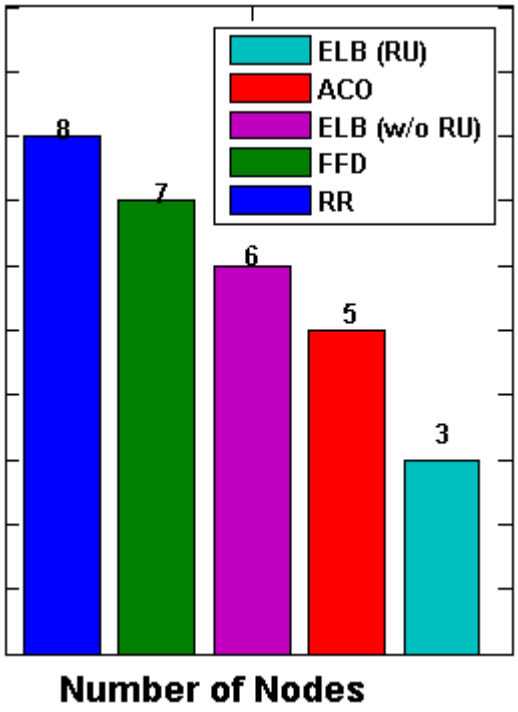


Figure 6.35: Number of Nodes used by each Technique

The Figure clearly depicts that the proposed ELB(w/o RU) technique uses less nodes than RR and FFD, but it uses more nodes than ACO and ELB(RU). The ELB(w/o RU) technique uses 25% less nodes than RR and 14.29% less nodes than FFD, whereas it is using 16.67% more nodes than ACO and 50% more nodes than ELB(RU). The reason for using less number of nodes by ELB(w/o RU) is that it is using FFO-EVMM migration technique to migrate the VMs from that active node whose energy consumption has reached a particular threshold value, thereby putting that node in sleep mode and saving energy. However, the reason for using more nodes is that ELB(w/o RU) does not consider the utilization of resources. The ELB(RU) approach is using the least number of nodes because of the use of ABC based ERU, that achieves optimal resource utilization, thereby reduces the number of the used nodes.

6.4.4 Test Case 4: Load Balancing

This subsection represents the principal task of load balancing by RR, FFD, ACO, ELB(RU) and ELB(w/o RU) techniques. In the proposed ELB(w/o RU) approach, every time, a new workload is processed by a node which is fulfilling its resource requirements. It is not checking the utilization of the resources like ELB(RU) approach, but it considers the performance as its main criterion and tries to achieve it by processing more and more number of workloads in a given period of time. Also, the ELB(w/o RU) approach uses FFO-EVMM technique to migrate the VMs to avoid hot-spots and to balance the system load on minimal required nodes thereby saving energy. The processed workloads and the servers used by all the techniques have been depicted through Figure 6.36. Further, this Figure shows the servers used by all the techniques before VM migrations are performed to balance the system load across all the active nodes except RR, as RR is not using any VM migration for balancing the load of the system.

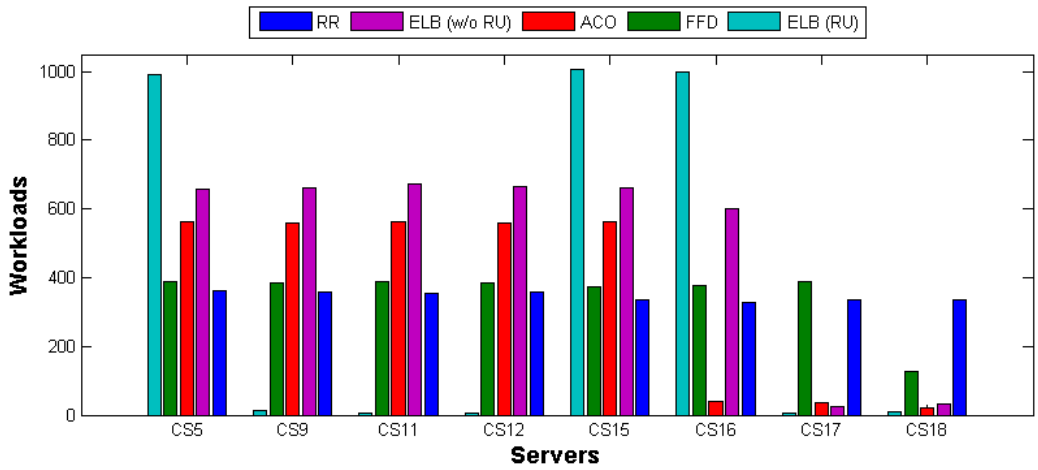


Figure 6.36: Servers Used by each Technique before VM Migrations

Figure 6.37 delineates the servers used by all the techniques after VM migrations thereby balancing the load of the system. The Figure also portrays that RR is not performing any VM migrations. The proposed ELB(w/o RU) approach uses less servers than RR & FFD but more servers than ACO and ELB(RU) and uses less VM migrations than FFD & ACO but more than ELB(RU) to balance the load of the system. This is because it is not considering the utilization of server's resources while assigning the workloads to them but it is using VM migrations to avoid over-utilization and under-utilization of servers, thereby aiding in dropping down the levels of the consumed energy.

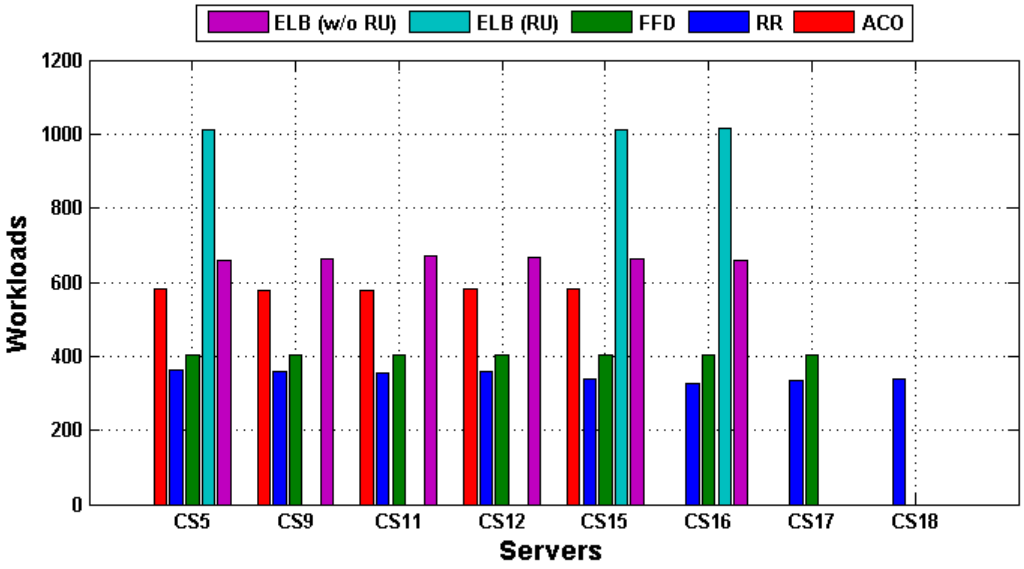


Figure 6.37: Servers used by each Technique after VM Migrations

6.4.5 Test Case 5: Energy Consumption

This subsection discusses about the energy consumed by RR, FFD, ACO, ELB(RU) & ELB(w/o RU) approaches as reducing the energy consumption of cloud data centers is the prime focus of this work. Figure 6.38 portrays that the energy consumption of the data center by using the ELB(w/o RU) technique is much less when compared to RR, FFD and ACO approaches but more than ELB(RU) approach. The proposed ELB(w/o RU) technique consumes an average of 29.43% less energy than the RR approach, 22.39% less energy than the FFD approach and 9.76% less energy than the ACO approach, but it consumes 40.64% more energy than the ELB(RU) approach. The reason for consuming less energy than RR & FFD is that it is saving more nodes and using less VM migrations than RR and FFD approaches. Also, ELB(w/o RU) consumes less energy than ACO, but it is using more nodes and less VM migrations than used by ACO. The reason for this is that ELB(w/o RU) is using energy-aware VM migrations to save nodes and the energy of the cloud data centers. However, ELB(w/o RU) consumes more energy than ELB(RU,) as ELB(RU) uses the least number of nodes and executes the lowest number of migrations, therefore consumes the minimum amount of energy.

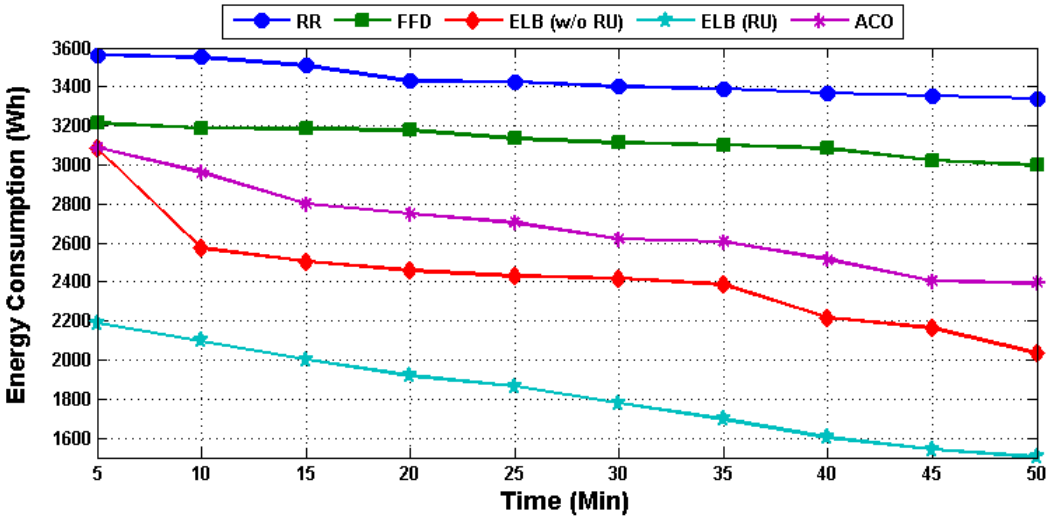


Figure 6.38: Energy Consumption by each Technique

6.4.6 Discussion

This subsection discusses the improvement of the proposed ELB(w/o RU) technique over RR, FFD, ACO & ELB(RU) approaches. Figure 6.39 illustrates that the ELB(w/o RU) approach handles the maximum number of workloads, when it is compared against the rest of the four approaches.

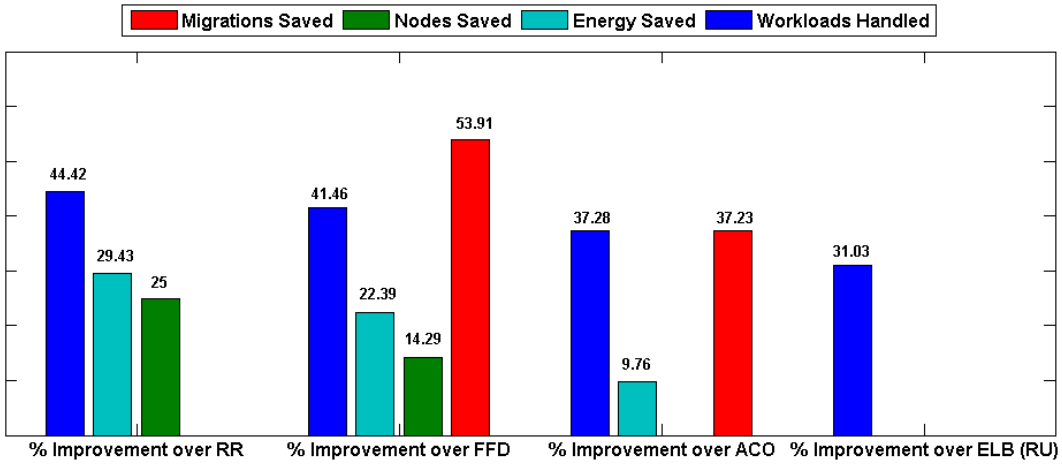


Figure 6.39: Percentage Improvement of ELB(w/o RU) Technique over Others

The Figure further depicts an improvement of ELB(w/o RU) over RR by showing 44.42% of average improvement in handling workloads, 29.43% & 25% enhancement in energy & node saving respectively. When compared to FFD , ELB(w/o RU) handles

on an average 41.46% more workloads, saves 14.29% nodes and 53.91% migrations, thereby aids in saving 22.39% more energy. The comparison of ELB(w/o RU) with ACO shows that ELB(w/o RU) handles 37.28% more workloads, curtails 37.23% of VM migrations and 9.76% of consumed energy. Finally, when ELB(w/o RU) is compared against ELB(RU) approach, it only shows an improvement in handling the workloads by 31.03% on an average. Overall, ELB(w/o RU) shows an average enhancement of 38.55% in the performance. It also reduces 20.53% of energy consumption by reducing 45.57% VM migrations and saving 19.64% nodes on an average.

Hence, it is clear from the above stated results that the proposed ELB(w/o RU) approach is better than the rest four RR, FFD, ACO, ELB(RU) techniques in improving performance by handling a large number of workloads, thereby satisfying more number of users by meeting a user satisfaction level. Also, this approach reduces the number of nodes by performing the required energy-aware VM migrations, thereby dropping the consumed energy levels too and making data center more energy-aware.

This section presented the verification and the implementation of the proposed ELB techniques in BSNL data center. The next section conduts the statistical analysis of the results.

6.5 Statistical Analysis of Results

This section employs a statistical method, namely the Coefficient of Variation (CoV), to analyze the statistical significance of the results. The CoV is a statistical measure of dispersion that depicts the amount of variability relative to the mean. To compare the dispersion of data sets having widely different means or different units, CoV is used in preference to the standard deviation.

It expresses the deviation of the data as a percentage of its average value, and is calculated as :

$$CoV = \frac{Std.Dev.}{AverageValue} \times 100 \tag{6.1}$$

where Std. Dev. is the standard deviation and Average Value is the mean. The CoV statistic offers an overall performance analysis of the technique used for generating the statistics. In Figures 6.40, 6.41 and 6.42, the CoV for the energy consumption, resource utilization and performance of each algorithm has been thoroughly examined. The range of CoV $(0.17\% - 1.76\%)$ for the energy consumption and $(1.59\% - 2.08\%)$ for the resource utilization approve the stability of the proposed ELB(RU) algorithm

as shown in Figures 6.40 and 6.41.

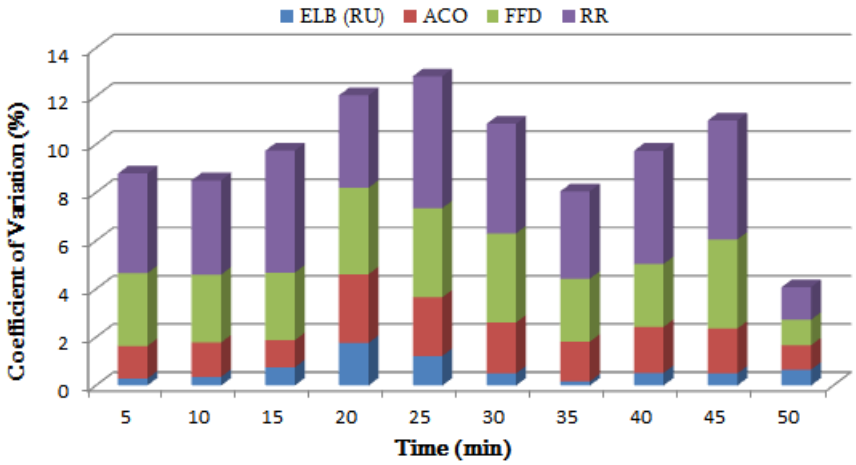


Figure 6.40: Coefficient of Variation for the Energy Consumption with each Technique

Whereas, the range of CoV (1.08%−2.14%) for the performance confirms the stability of the proposed ELB(w/o RU) algorithm as shown in Figure 6.42. As the value of CoV is small, it means that the proposed Energy-aware Load Balancing algorithms are more efficient in balancing the load of the cloud. As, the system with small CoV value is more stable, so the proposed algorithms achieved the best results in the cloud data center with regard to the energy consumption, resource utilization and performance.

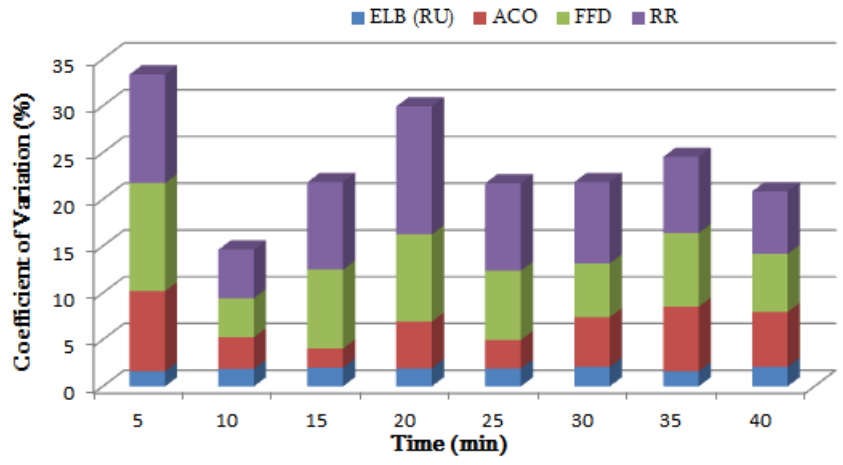


Figure 6.41: Coefficient of Variation for the Resource Utilization with each Technique

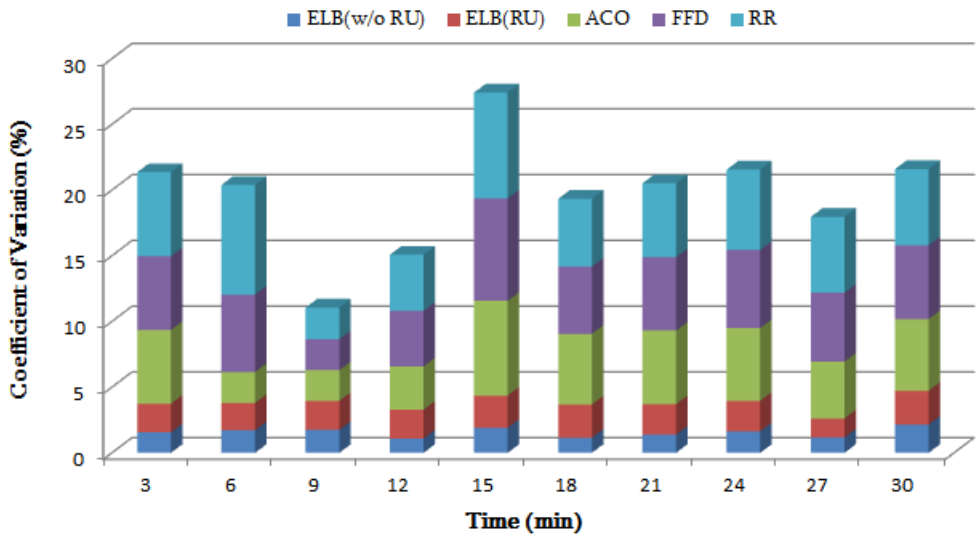


Figure 6.42: Coefficient of Variation for the Performance with each Technique

6.6 Conclusion

This chapter presented the verification details, experimental setup and testing results of the proposed ELB techniques in BSNL data center. The experimental results demonstrate the efficacy of ELB(RU) by saving on an average, 49.68% CPU utilization, 24.41% memory utilization, 53.21% nodes and reducing 63.10% VM migrations, thereby curtailing 40.47% of the average energy consumption. Also, the efficiency of ELB(w/o RU) is depicted by 38.55% performance enhancement, 20.53% reduction in the energy consumption, 45.57% reduction in the VM migrations and 19.64% reduction in the nodes on an average.

The next chapter concludes the thesis by highlighting the main contributions of this research work and gives the future research directions.

Chapter 7

Conclusions and Future Directions

Recently, energy efficiency has appeared as the utmost essential design requirements for the current computing paradigms. It extends from single servers to data centers and the clouds, as they consume massive volumes of electrical power. For this reason, an effectual energy management is particularly essential for the cloud data centers.

This thesis focuses upon various issues, including low resource utilization levels, the number of VM migrations, load balancing and the energy management in the cloud environments. An ABC based ERU model has been proposed to address the issue of low resource utilization levels. An FFO based EVMM technique has been presented to address the issue of VM migrations. A novel ELB technique, that uses an amalgamation of ERU and FFO-EVMM, has been presented to balance the load of the cloud across minimum required nodes, while trying to maximize the energy efficiency through the efficient usage of cloud resources. Another ELB technique has also been presented, that enhances the system performance to a large extent and curtails the energy consumption through energy-aware VM migrations thereby reducing the number of nodes.

In this final chapter, concluding remarks on this research work have been given, by describing the advancement, made towards the objectives of the research, in terms of developing energy-aware load balancing techniques, to balance the load and to save energy. The chapter details the outcome of each chapter and later highlights the contributions of this research work. Furthermore, it discusses the directions for future research.

7.1 Conclusions

The aim of this research work has been to design and develop the Energy-aware Load Balancing (ELB) techniques for cloud computing, which has been addressed by the proposed Energy-aware Load Balancing (ELB) model for cloud computing. As investigated, the efficient resource utilization and VM migration are the important factors in bringing down the energy consumption levels in the cloud data centers. Hence, the proposed ELB techniques are based on these two aspects. The first aspect is focused in Chapter 3 and the second aspect is focused in Chapter 4.

It has been demonstrated as a part of the Literature Review in Chapter 2, that the resource utilization, VM migration and load balancing are the indispensable parts of the cloud environment for handling the energy management issue. Furthermore, cloud service providers are always confronted with the resource under-utilization and energy management issues to support green computing. After an extensive survey of the existing approaches related to the resource utilization, VM migration and load balancing, the chapter figures out their limitations and a need to develop the Energy-aware Load Balancing techniques with proper resource utilization and VM migration.

Chapter 3 has dealt with the first aspect of utilizing the resources efficiently. It has proposed an Energy-aware Resource Utilization (ERU) model that is based on the Artificial Bee Colony (ABC) optimization technique. The key objectives, requirements, architecture and the components of the proposed ERU model have been discussed in detail followed by its detailed design, algorithm & the flowchart.

This model efficiently manages the cloud resources and enhances their utilization to conserve energy. To minimize the energy consumption and to avoid the degradation of the performance, the model considers the conflicts between the utilization levels of the CPU and the memory, as well as the types of the workloads. The consolidation of the workloads have been done carefully to avoid the contention of the resources. Various thresholds have been set to assign a workload to a node. The ERU model assigns the workloads to the nodes to save energy and preserve the performance. The verification and the validation of the proposed ERU model has been done through the CloudSim toolkit. The results included in this chapter, demonstrate the effectiveness of the ERU model in utilizing the resources efficiently, thereby reducing the energy consumption levels in the cloud data centers. Once the resources are utilized effectively, the challenge is to migrate the VMs to the most energy-aware nodes, to balance the load across minimum required nodes, and to save energy in the cloud data centers.

Obtaining energy optimization through VM migration by regulating workloads on individual nodes is an NP-hard problem and heuristic methods are often used to resolve such kind of problems. In the scenario of energy consumption by VM migration in a dynamic cloud environment, a linear model based on FireFly Optimization (FFO) is formulated that runs an FFO algorithm, which is able to solve the energy consumption issue with the attraction property of fireflies. The capability of fireflies to get attracted towards the brighter fireflies is the basis for considering the FFO. Chapter 4 has dealt with the second aspect of VM Migration Technique. It has devised an FFO based Energy-aware Virtual Machine Migration (FFO-EVMM) technique that effectively maps the most-loaded VMs to the most energy-aware nodes, hence maximizes the energy-efficiency through the optimum migration of VMs, thereby improving resource utilization levels. The FFO-EVMM technique has also been verified and validated through extensive simulations performed in the CloudSim simulator. The efficacy of the proposed technique has been approved by the results, given in this chapter.

An Energy-aware Load Balancing (ELB) model has been presented and two corresponding ELB techniques have been proposed in Chapter 5. One ELB technique is with Resource Utilization (ELB(RU)), which is an amalgamation of ERU & FFO-EVMM presented in Chapter 3 & Chapter 4 respectively. The ELB(RU) technique together with ERU & FFO-EVMM approaches, aim to balance the workload across the minimum number of active servers by the least number of VM migrations, thereby achieving load balancing & maximum resource utilization, hence improving the energy-efficiency of the cloud data centers, without degrading the performance levels. The other ELB technique is without Resource Utilization (ELB(w/o RU)), that enhances the system performance level by handling the maximum number of workloads. The ELB(w/o RU) technique uses only FFO-EVMM approach to make energy-aware VM migration decision, to save energy and to reduce the number of nodes. The ELB model helps in making a decision about one of the two techniques, to be followed for the load balancing.

The ELB techniques have been empirically evaluated by executing them in the data center of BSNL, Chandigarh. Chapter 6 provides the implementation details and shows the experimental results obtained for validating these techniques. The impact of the proposed ELB techniques has been evaluated in BSNL data center and the experimental results obtained have been discussed from various perspectives such as effect of resource utilization levels, number of hosts used, number of VM migrations performed, number of workloads, energy consumption levels, etc. Also, the results obtained have

been analyzed statistically in this chapter, to prove the stability of the proposed ELB techniques in balancing the load of the cloud data centers. The precise contributions of this thesis are listed below:

- An extensive literature survey of the work done in the area of resource utilization, VM migration and load balancing and addresses the challenges such as Optimum Resource Utilization and VM Migration to significantly reduce the Energy Consumption
- A bio-inspired Artificial Bee Colony (ABC) based Energy-aware Resource Utilization (ERU) model that increases the utilization of server resources, reduces the energy consumption and heat dissipation, hence contributes directly to green computing
- A bio-inspired Firefly Optimization based Energy-aware VM Migration (FFO-EVMM) technique that maximizes the energy-efficiency through the optimum migration of VMs, thereby improving resource utilization levels
- Two Energy-aware Load Balancing techniques with & without Resource Utilization, namely ELB(RU) and ELB(w/o RU) to balance the system load by enhancing the resource utilization and performance levels, thereby minimizing the energy consumption of the cloud data centers
- The proposed ELB algorithms have considered the perspectives of both the cloud provider and the cloud user, such as resource utilization, energy consumption and performance
- The verification and validation of the scalability and efficacy of the proposed ERU & FFO-EVMM, using the CloudSim simulator
- A case study at BSNL data center, Chandigarh, to evaluate and analyze the performance of the proposed ELB algorithms
- All the techniques reduce energy consumption, thereby indirectly reduce carbon emissions and cooling requirements of the cloud data centers, to help achieve green computing.

7.2 Future Research Directions

The energy-aware resource utilization technique, the energy-aware VM migration technique and the energy-aware load balancing techniques can be enhanced for future research. Some of the research directions are outlined below :

- It would be exciting to examine the possible integration of all the techniques with various other QoS parameters like cost, fault tolerance and reliability etc. which are also significant for cloud computing. Also, the issues of security, privacy and compliance management still remain important in a distributed computing environment, therefore, all the proposed techniques can be augmented to support these attributes too.

- The proposed techniques address the resource utilization and energy management issues in cloud computing. As a part of future research, other resource management issues such as resource monitoring and auditing can also be considered as significant challenges in a business driven cloud computing environment.

- Besides, all the proposed techniques can be enhanced to support disk-intensive and network workloads along with the network and the storage resources and can be tested with an increased data set and more number of servers to achieve an enhanced efficiency.

- In this work, reducing energy consumption indirectly reduces carbon emissions and cooling requirements. But the direct measurement of extent of reduction with some metrics has not been considered in the present work. These metrics would highlight the impact of the work done by including direct readings in future for more accurate measurements of energy savings in clouds.

Bibliography

[1] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley*, vol. 28, no. 13, pp. 1–25, 2009. http://home.cse.ust.hk/ weiwa/teaching/Fall15 - COMP6611B/reading_list/AboveTheClouds.pdf.

[2] G. Pallis, "Cloud computing: the new frontier of internet computing," *IEEE Internet Computing, IEEE Computer Society*, vol. 14, no. 5, pp. 70–73, 2010.

[3] K. T. Rao, P. S. Kiran, and L. S. S. Reddy, "Energy efficiency in datacenters through virtualization: A case study," *Global Journal of Computer Science and Technology*, vol. 10, no. 3, 2010.

[4] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*, pp. 826–831, IEEE Computer Society, 2010. https://dl.acm.org/purchase.cfm?id=1845139 &CFID=828049556&CFTOKEN=22335400.

[5] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review, ACM*, vol. 39, no. 1, pp. 50–55, 2008.

[6] M. A Vouk, "Cloud computing–issues, research and implementations," *CIT. Journal of Computing and Information Technology, SRCE-Sveučilišni računski centar*, vol. 16, no. 4, pp. 235–246, 2008.

[7] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers," *Journal of Parallel and Distributed Computing, Elsevier*, vol. 71, no. 6, pp. 732–749, 2011.

[8] R. W. Lucky, "Cloud computing [reflections]," *IEEE Spectrum, IEEE*, vol. 46, no. 5, pp. 27–45, 2009.

[9] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 10–13, 2009.

[10] A. Singh and M. Hemalatha, "Cloud computing for academic environment," *International Journal of Information and Communication Technology Research, Citeseer*, vol. 2, no. 2, pp. 97–101, 2012.

[11] I. Sriram and A. Khajeh-Hosseini, "Research agenda in cloud technologies," *arXiv preprint arXiv:1001.3259*, 2010. https://arxiv.org/ftp/arxiv/papers/1001/1001.3259.pdf.

[12] G. Kecskemeti, M. Gergely, A. Visegradi, Z. Nemeth, J. Kovacs, and P. Kacsuk, "One click cloud orchestrator: Bringing complex applications effortlessly to the clouds," in *European Conference on Parallel Processing*, pp. 38–49, Springer, 2014. http://link.springer.com/chapter/10.1007/978-3-319-14313-2_4.

[13] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications, Springer*, vol. 1, no. 1, pp. 7–18, 2010.

[14] S. K. Chowdhary, A. Yadav, and N. Garg, "Cloud computing: Future prospect for e-health," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 3, pp. 297–299, IEEE, 2011.

[15] F. Teng, *Management of Data and Scheduling of Tasks on Architecture Distributees*. PhD thesis, Ph. D. thesis. École Centrale: A University Institution, Paris, 2011.

[16] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, pp. 44–51, Ieee, 2009. http://s3.amazonaws.com/academia.edu.documents/34625815/A_Taxonomy_and_Survey_of_Cloud_Computing_System.pdf.

[17] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing, Springer*, vol. 28, no. 2, pp. 137–146, 2010. http://dx.doi.org/10.1007/s00354-008-0081-5.

[18] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," *Georgia Institute of Technology, Tech. Rep*, p. 138, 2009. http://www.cercs.gatech.edu/tech-reports/tr2009/ git-cercs-09-13.pdf.

[19] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg*, vol. 53, no. 6, p. 50, 2010.

[20] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.

[21] P. T. Jaeger, J. Lin, J. M. Grimes, and S. N. Simmons, "Where is the cloud? geography, economics, environment, and jurisdiction in cloud computing," *First Monday*, vol. 14, no. 5, 2009.

[22] T. Ercan, "Effective use of cloud computing in educational institutions," *Procedia-Social and Behavioral Sciences, Elsevier*, vol. 2, no. 2, pp. 938–942, 2010. http://dx.doi.org/10.1016/j.sbspro.2010.03.130.

[23] L. Mei, W. K. Chan, and T. Tse, "A tale of clouds: paradigm comparisons and some thoughts on research issues," in *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, pp. 464–469, Ieee, 2008. http://dx.doi.org/10.1109/APSCC.2008.168.

[24] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[25] A. K. Coskun and T. S. Rosing, "Improving energy efficiency and reliability through workload scheduling in high-performance multicore processors." http://vote.dvcon.com/sites/default/files/ COSKUN-MPSOCREL_Posted.pdf.

[26] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review, ACM*, vol. 39, no. 1, pp. 68–73, 2008. http://dx.doi.org/10.1145/1496091.1496103.

[27] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, vol. 9, 2011. http://www.missioncriticalmagazine.com/ext/resources /MC/Home/Files/PDFs/Koomey_Data_Center.pdf.

[28] M. Blazek, H. Chong, W. Loh, and J. G. Koomey, "Data centers revisited: assessment of the energy impact of retrofits and technology trends in a high-density computing facility," *Journal of Infrastructure Systems, American Society of Civil Engineers*, vol. 10, no. 3, pp. 98–104, 2004. http://dx.doi.org/10.1061/(ASCE)1076-0342.

[29] K. J. Christensen, C. Gunaratne, B. Nordman, and A. D. George, "The next frontier for communications networks: power management," *Computer Communications, Elsevier*, vol. 27, no. 18, pp. 1758–1770, 2004. http://dx.doi.org/10.1016/j.comcom.2004.06.012.

[30] A. Ogasawara, "Energy issues confronting the information and communications sector-need to reduce the power consumed by the communications infrastructure," 2006. http://www.nistep.go.jp/achiev/ftx/eng/stfc/stt021e/qr21pdf/STTqr2102.pdf.

[31] "International energy outlook," 2013. http://www.eia.gov/forecasts/ieo/pdf/0484(2013).pdf.

[32] Envantage, "World carbon dioxide emissions by country," 2013. http://www.energy-awareness-training.co.uk/world-carbon-dioxideemissions-by-country.

[33] T. Kaur and I. Chana, "Energy efficiency techniques in cloud computing: A survey and taxonomy," *ACM Computing Surveys (CSUR), ACM*, vol. 48, no. 2, p. 22, 2015.

[34] T. C. Group, "Smart 2020: Enabling the low power economy in the information age," 2008. http://www.smart2020.org/_assets/files/02 _smart2020Report.pdf.

[35] L. Johnsson, "Overview of data centers energy efficiency evolution," in *Handbook of Energy Aware and Green Computing* (S. R. Ishfaq Ahmad, ed.), vol. 2, ch. 43, CRC Press, Taylor & Francis Group, 2012.

[36] M. K. Patterson, D. Costello, P. Grimm, and M. Loeffler, "Data center tco; a comparison of high-density and low-density spaces," *Thermal Challenges in Next Generation Electronic Systems (THERMES 2007)*, pp. 42–53, 2007. http://www.mm4m.net/library/datacenter_TCO _low_high_density.pdf.

[37] S. Albers, "Energy-efficient algorithms," *Communications of the ACM, ACM*, vol. 53, no. 5, pp. 86–96, 2010. http://doi.acm.org/10.1145/1735223.1735245.

[38] A. Weiss, "Computing in the clouds," *Computing*, vol. 16, 2007.

[39] E. Mobil, "The outlook for energy: A view to 2040," *ExxonMobil: Irving, TX, USA*, 2013. http://www.exxonmobil.com/Corporate/Files/news_pub_2013eo_us.pdf.

[40] Envantage, "New report reveals warehouses overspend on energy by 190m," 2013. http://www.en-mat.com/newreport-reveals-, warehouses-overspend-on-energy-by-190m.

[41] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, *et al.*, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in computers, Academic Press*, vol. 82, no. 2, pp. 47–111, 2011.

[42] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers.," in *USENIX annual technical conference, General Track*, pp. 61–75, 2005. https://www.usenix.org/legacy/events/usenix05/tech/general/full_papers/moore/moore.pdf.

[43] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power," in *Design Automation, 1993. 30th Conference on*, pp. 74–79, IEEE, 1993. http://dx.doi.org/10.1109/DAC.1993.203922.

[44] A. Plepys, "The grey side of ict," *Environmental Impact Assessment Review, Elsevier*, vol. 22, no. 5, pp. 509–523, 2002. http://dx.doi.org/10.1016/S0195-9255(02)00025-2.

[45] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The computer journal, Br Computer Soc*, vol. 53, no. 7, pp. 1045–1051, 2010.

[46] Cloudcommons, "Top 10 apps for cloud, private cloud implementation issues," 2012. http://cloudcomputing.sys con.com/node/1653265.

[47] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 27–33, Ieee, 2010. http://dx.doi.org/10.1109/AINA.2010.187.

[48] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computingthe business perspective," *Decision support systems, Elsevier*, vol. 51, no. 1, pp. 176–189, 2011. http://dx.doi.org/10.1016/j.dss.2010.12.006.

[49] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole, "Energy-efficient application-aware online provisioning for virtualized clouds and data centers," in *Green Computing Conference, 2010 International*, pp. 31–45, IEEE, 2010. http://dx.doi.org/10.1109/GREENCOMP.2010.5598283.

[50] F. Owusu and C. Pattinson, "The current state of understanding of the energy efficiency of cloud computing," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom12)*, pp. 1948–1953, IEEE, 2012. http://dx.doi.org/10.1109/TrustCom.2012.270.

[51] J. Chase, G. C. Fox, R. J. Figueiredo, A. Grimshaw, P. Watson, and M. Yousif, "Thoughts on the state of cloud over the next five years," *IEEE Cloud Computing*, vol. 2, no. 1, pp. 26–40, 2014.

[52] S. Sahni and V. Varma, "A hybrid approach to live migration of virtual machines," in *2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, vol. 6, 2012. file:///C:/Users/Microsoft/Downloads/sahni_ccem2012.pdf.

[53] T. Kaur and I. Chana, "Energy conscious allocation and scheduling of tasks in ict cloud paradigm," in *Proceedings of International Conference on ICT for Sustainable Development*, pp. 589–601, Springer, 2016.

[54] T. Kaur and I. Chana, "Energy aware scheduling of deadline-constrained tasks in cloud computing," *Cluster Computing, Springer*, vol. 19, no. 3, pp. 1–20, 2016. doi:10.1007/s10586-016-0566-9.

[55] T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pp. 88–94, IEEE, 2010. http://vmbichler6.informatik.tu-muenchen.de/ files/bichler-research/2010_setzer _svdreassignment.pdf.

[56] B. Yagoubi and M. Meddeber, "Distributed load balancing model for grid computing," *ARIMA journal*, vol. 12, pp. 43–60, 2010.

[57] B. Yagoubi and Y. Slimani, "Task load balancing strategy for grid computing," *Journal of Computer Science*, vol. 3, no. 3, pp. 186–194, 2007.

[58] A. M. Alakeel, "A guide to dynamic load balancing in distributed computer systems," *International Journal of Computer Science and Information Security, Citeseer*, vol. 10, no. 6, pp. 153–160, 2010.

[59] B. Yagoubi and Y. Slimani, "Dynamic load balancing strategy for grid computing," *Transactions on Engineering, Computing and Technology*, vol. 13, pp. 260–265, 2006.

[60] K. Y. Kabalan, W. W. Smari, and J. Y. Hakimian, "Adaptive load sharing in heterogeneous systems: Policies, modifications, and simulation," *International Journal of Simulation, Systems, Science and Technology, Citeseer*, vol. 3, no. 1-2, pp. 89–100, 2002.

[61] K. Lu, "Decentralized load balancing in heterogeneous computational grids," 2007. rp-www.cs.usyd.edu.au/research/2008 _Kai_Lu_thesis.pdf.

[62] A. Chhabra, G. Singh, S. S. Waraich, B. Sidhu, and G. Kumar, "Qualitative parametric comparison of load balancing algorithms in parallel and distributed computing environment," *Word Academy of Science, Engineering and Technology*, pp. 39–42, 2006. http://www.waset.org/publications/4737.

[63] J. A. Baikerikar, S. K. Surve, and S. U. Prabhu, "Comparison of load balancing algorithms in a grid," in *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, pp. 20–23, IEEE, 2010. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5452621.

[64] M. Wegdam, *Dynamic reconfiguration and load distribution in component middleware*. Telematics Institute, 2003. http://doc.utwente.nl/41469/1/t0000025.pdf.

[65] L. W. Zaki M. J. and P. S., "Customized dynamic load balancing for a network of workstations," in *High performance computing, 2002 IEEE conference on, NY, USA*, pp. 156–162, IEEE, 2002. https://www.researchgate.net/profile/Mohammed_Zaki4/publication

/222451490_Customized_Dynamic_Load_Balancing_for_a_Network_of_Workstations /links/54296d380cf2e4ce940e691b.pdf.

[66] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing, Springer*, vol. 60, no. 2, pp. 268–280, 2012.

[67] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286, USENIX Association, 2005. https://dl.acm.org/purchase.cfm?id=1251223&CFID=828049556 &CFTOKEN=22335400.

[68] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," *INC, IMS and IDC*, pp. 44–51, 2009. http://s3.amazonaws.com/academia.edu.documents/ 34625815/A_Taxonomy_and_Survey_of_Cloud_Computing _System.pdf.

[69] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10, pp. 1–5, San Diego, California, 2008.

[70] Y. Song, Y. Zhang, Y. Sun, and W. Shi, "Utility analysis for internet-oriented server consolidation in vm-based data centers," in *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1–10, IEEE, 2009. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.360.8008 &rep=rep1&type=pdf.

[71] I. Ahmad and S. Ranka, *Handbook of energy-aware and green computing*, vol. 1. CRC Press, 2012.

[72] R. Mata-Toledo and P. Gupta, "Green data center: how green can we perform?," *Journal of Technology Research, Academic and Business Research Institute (AABRI)*, vol. 2, no. 1, pp. 1–8, 2011.

[73] S. Kabiraj, V. Topkar, and R. Walke, "Going green: a holistic approach to transform business," *arXiv preprint arXiv:1009.0844*, p. 110, 2010. https://arxiv.org/ftp/arxiv/papers/ 1009/1009.0844.pdf.

[74] A. Auweter, A. Bode, M. Brehm, H. Huber, and D. Kranzlmüller, "Principles of energy efficiency in high performance computing," in *International Conference on Information and Communication on Technology*, pp. 18–25, Springer, 2011. http://link.springer.com/chapter/10.1007%2F978-3-642-23447-7_3.

[75] K. Nagothu, B. Kelley, J. Prevost, and M. Jamshidi, "Ultra low energy cloud computing using adaptive load prediction," in *World Automation Congress (WAC), 2010*, pp. 1–7, IEEE, 2010. http://engineering.utsa.edu/ bkelley/Pdf /Final%20WAC.pdf.

[76] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation," in *Industrial Mechatronics and Automation (ICIMA), 2010 2nd International Conference on*, vol. 2, pp. 240–243, IEEE, 2010.

[77] M. Yue, "A simple proof of the inequality ffd (l) 11 9 opt (l)+ 1, l for the ffd bin-packing algorithm. report no. 90665," *Research Institute for Discrete Mathematics, University of Bonn*, 1990. http://compalg.inf.elte.hu/ tony/ Kutatas/BinPacking/Yue-FFD-bound-ActaSinica -1991.pdf.

[78] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems, Elsevier*, vol. 43, no. 2, pp. 73–81, 1997.

[79] S. Mahambre, P. Kulkarni, U. Bellur, G. Chafle, and D. Deshpande, "Workload characterization for capacity planning and performance management in iaas cloud," in *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, pp. 1–7, IEEE, 2012. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6354624.

[80] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy, survey, and issues of cloud computing ecosystems," in *Cloud Computing*, pp. 21–46, Springer, 2010. http://link.springer.com/chapter/10.1007/978-1-84996-241-4_2.

[81] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *INFOCOM, 2012 Proceedings IEEE*, pp. 702–710, IEEE, 2012. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6195815.

[82] H. Mehta, P. Kanungo, and M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environments," in *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pp. 370–375, ACM, 2011. https://dl.acm.org/purchase.cfm?id=1980102 &CFID=828049556&CFTOKEN=22335400.

[83] A. M. Nakai, E. Madeira, and L. E. Buzato, "Load balancing for internet distributed services using limited redirection rates," in *Dependable Computing (LADC), 2011 5th Latin-American Symposium on*, pp. 156–165, IEEE, 2011. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5783395.

[84] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services," *Performance Evaluation, Elsevier*, vol. 68, no. 11, pp. 1056–1071, 2011.

[85] X. Liu, L. Pan, C.-J. Wang, and J.-Y. Xie, "A lock-free solution for load balancing in multi-core environment," in *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pp. 1–4, IEEE, 2011. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5873313.

[86] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *2010 3rd International symposium on parallel architectures, algorithms and programming*, pp. 89–96, IEEE, 2010. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5715067.

[87] A. Bhadani and S. Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud," in *Proceedings of the Third Annual ACM Bangalore Conference*, p. 16, ACM, 2010. https://www.researchgate.net/profile/Sanjay_Chaudhary3/publication/221233022_Performance_evaluation_of_web_servers_using_central_load_balancing_policy_over_virtual_machines_on_cloud/links/00b7d533e6ba8d4014000000.pdf.

[88] H. Liu, S. Liu, X. Meng, C. Yang, and Y. Zhang, "Lbvs: A load balancing strategy for virtual storage," in *2010 International Conference on Service Sciences*, pp. 257–262, IEEE, 2010. http://osnet.cs.nchu.edu.tw/powpoint/seminar/2010/LBVS_A%20Load%20Balancing%20Strategy%20for%20Virtual %20Storage.pdf.

[89] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in *International Conference on Web Information Systems and Mining*, pp. 271–277, Springer, 2010. http://link.springer.com/chapter/10.1007/978-3-642-16515-3_34.

[90] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior, SAGE Publications*, vol. 12, no. 3-4, pp. 223–240, 2004.

[91] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, pp. 551–556, IEEE, 2010. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5480636.

[92] M. Randles, E. Odat, D. Lamb, O. Abu-Rahmeh, and A. Taleb-Bendiab, "A comparative experiment in distributed load balancing," in *Developments in eSystems Engineering (DESE), 2009 Second International Conference on*, pp. 258–265, IEEE, 2009. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5395118.

[93] O. A. Rahmeh, P. Johnson, and A. Taleb-Bendiab, "A dynamic biased random sampling scheme for scalable and reliable grid networks," *INFOCOMP Journal of Computer Science*, vol. 7, no. 4, pp. 1–10, 2008.

[94] F. Saffre, R. Tateson, J. Halloy, M. Shackleton, and J. L. Deneubourg, "Aggregation dynamics in overlay networks and their implications for self-organized distributed applications," *The Computer Journal, Br Computer Soc*, vol. 52, no. 4, pp. 397–412, 2009.

[95] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 1, pp. 108–113, IEEE, 2010.

[96] V. Nae, R. Prodan, and T. Fahringer, "Cost-efficient hosting and load balancing of massively multiplayer online games," in *2010 11th IEEE/ACM International Conference on Grid Computing*, pp. 9–16, IEEE, 2010. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5697956.

[97] R. Stanojevic and R. Shorten, "Load balancing vs. distributed rate limiting: an unifying framework for cloud control," in *2009 IEEE International Conference on Communications*, pp. 1–6, IEEE, 2009. http://eprints.maynoothuniversity.ie/2044/1/RS_DRLvsLB_ICC2009.pdf.

[98] Y. Zhao and W. Huang, "Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pp. 170–175, IEEE, 2009. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5331732.

[99] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, p. 53, IEEE Press, 2008. http://www.cs.cmu.edu/ chensm/Big_Data_reading_group/papers/ ServerStorageVirt-SinghKM.pdf.

[100] M. K. Megharaj G.C., "Two level hierarchical model of load balancing in cloud," *International Journal of Emerging Technology and Advanced Engineering (IJETAE)*, vol. 3, no. 10, pp. 307–311, 2013.

[101] S. C. Izain Nurfateha Ruzan and P. Razmara, "A hybrid algorithm using genetic algorithm–hadoop mapreduce optimization for energy efficiency in cloud computing platform," *International Journal of Science and Research (IJSR)*, vol. 3, no. 11, p. 16301641, 2014. http://www.ijsr.net/archive/v3i11/T0NUMTQxMjUw.pdf.

[102] T. Anandharajan and M. Bhagyaveni, "Co-operative scheduled energy aware load-balancing technique for an efficient computational cloud," *International Journal of Computer Science, Citeseer*, no. 8, pp. 571–576, 2011. http://citeseerx.ist.psu.edu/viewdoc /download?doi=10.1.1.402.9531&rep=rep1 &type=pdf#page=592.

[103] J. M. Galloway, K. L. Smith, and S. S. Vrbsky, "Power aware load balancing for cloud computing," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, pp. 19–21, 2011.

[104] J. Adhikari and S. Patil, "Double threshold energy aware load balancing in cloud computing," in *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*, pp. 1–6, IEEE, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber =6726664.

[105] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-mmt: A load balancing method for power consumption management in cloud computing," in *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pp. 76–80, IEEE, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber =6612165.

[106] P. Dalapati and G. Sahoo, "Green solution for cloud computing with load balancing and power consumption management," *International Journal Of Emerging Technology and Advanced Engineering (ISSN: 2250-2459), Citeseer*, vol. 3, no. 3, 2013. http://citeseerx.ist.psu.edu/viewdoc /download?doi=10.1.1.413.5509&rep=rep1&type=pdf.

[107] N. M. Al Sallami, "Load balancing in green cloud computation," in *Proceedings of the World Congress on Engineering*, vol. 2, 2013.

[108] N. M. Al Sallami and S. A. Al Alousi, "Load balancing with neural network," *(IJACSA) International Journal of Advanced Computer Science and Applications, Citeseer*, vol. 4, no. 10, 2013.

[109] S. RamKumar, V. Vaithiyanathan, and M. Lavanya, "Towards efficient load balancing and green it mechanisms in cloud environment," *World Applied Sciences Journal*, no. 29, pp. 159–165, 2014. http://www.idosi.org/wasj/wasj29(dmsct)14/30.pdf.

[110] S. Singh and I. Chana, "Qos-aware autonomic resource management in cloud computing: a systematic review," *ACM Computing Surveys (CSUR), ACM*, vol. 48, no. 3, p. 42, 2016.

[111] M. Bichler, T. Setzer, and B. Speitkamp, "Capacity planning for virtualized servers," in *Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA*, 2006. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1025862.

[112] M. Chebiyyam, R. Malviya, S. K. Bose, and S. Sundarrajan, "Server consolidation: Leveraging the benefits of virtualization," *SETLabs Briefings, Infosys Labs*, vol. 7, no. 1, pp. 65–74, 2009.

[113] http://searchdatacenter.techtarget.com/sDefinitions/0,,sid80_gci1070280,00.html.

[114] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on services computing, IEEE*, vol. 3, no. 4, pp. 266–278, 2010.

[115] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of Grid Computing, Springer*, vol. 14, no. 2, pp. 217–264, 2016.

[116] W. Vogels, "Beyond server consolidation," *Queue, ACM*, vol. 6, no. 1, pp. 20–26, 2008.

[117] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: Coordinated multi-level power management for the data center," in *ACM SIGARCH Computer Architecture News*, vol. 36, pp. 48–59, ACM, 2008.

[118] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks, Elsevier*, vol. 53, no. 17, pp. 2923–2938, 2009.

[119] R. Nathuji and K. Schwan, "Virtualpower: coordinated power management in virtualized enterprise systems," in *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 265–278, ACM, 2007.

[120] S. Singh and I. Chana, "Energy based efficient resource scheduling: a step towards green computing," *Int J Energy Inf Commun*, vol. 5, no. 2, pp. 35–52, 2014.

[121] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems, Elsevier*, vol. 25, no. 6, pp. 599–616, 2009.

[122] S. Singh and I. Chana, "Cloud resource provisioning: survey, status and future research directions," *Knowledge and Information Systems, Springer*, pp. 1–65, 2015. link.springer.com/article/10.1007/s10115-016-0922-3.

[123] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems, Elsevier*, vol. 28, no. 5, pp. 755–768, 2012.

[124] M. Sharifi, H. Salimi, and M. Najafzadeh, "Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques," *The Journal of Supercomputing, Springer*, vol. 61, no. 1, pp. 46–66, 2012.

[125] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–128, IEEE, 2007. citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.8744&rep=rep1&type=pdf.

[126] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-tiered on-demand resource scheduling for vm-based data center," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 148–155, IEEE Computer Society, 2009. https://www.researchgate.net/profile/

Yuzhong_Sun/publication/220940956_Multi-Tiered_On-Demand_Resource_Scheduling_for_VM-Based_Data_Center/links/09e4150865654dc72d000000.pdf.

[127] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*, pp. 327–334, IEEE, 2009. http://citeseerx.ist.psu.edu/viewdoc/ download?doi=10.1.1.173.350&rep=rep1&type=pdf.

[128] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing, Springer*, vol. 12, no. 1, pp. 1–15, 2009.

[129] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308*, 2010. https://arxiv.org/ftp/arxiv/papers/1006/1006.0308.pdf.

[130] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, pp. 26–33, IEEE Computer Society, 2011. https://hal.inria.fr/file/index/ docid/594992/filename/RR-7622.pdf.

[131] T. Rings, G. Caryer, J. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, and I. Stokes-Rees, "Grid and cloud computing: opportunities for integration with the next generation network," *Journal of Grid Computing, Springer*, vol. 7, no. 3, pp. 375–393, 2009.

[132] C. Min, I. Kim, T. Kim, and Y. I. Eom, "Vmmb: virtual machine memory balancing for unmodified operating systems," *Journal of Grid Computing, Springer*, vol. 10, no. 1, pp. 69–84, 2012.

[133] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 577–578, IEEE, 2010. http://www.buyya.com/ raj/papers/ EnergyEfficientVMAllocationCCGrid2010.pdf.

[134] I. Rodero, H. Viswanathan, E. K. Lee, M. Gamell, D. Pompili, and M. Parashar, "Energy-efficient thermal-aware autonomic management of virtualized hpc cloud infrastructure," *Journal of Grid Computing, Springer*, vol. 10, no. 3, pp. 447–473, 2012.

[135] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient vm scheduling for cloud data centers: exact allocation and migration algorithms," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 671–678, IEEE, 2013. https://www.researchgate.net/profile/Makhlouf_Hadji/publication/261093539_Energy_Efficient_VM_Scheduling_for_Cloud_Data_

Centers_Exact_Allocation_and_Migration_Algorithms/links/547229c50cf24af340c531b9.pdf.

[136] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 482–489, IEEE Computer Society, 2012. https://hal.archives-ouvertes.fr/file/ index/docid/664621/filename/efeller.pdf.

[137] M. Tarighi, S. A. Motamedi, and S. Sharifian, "A new model for virtual machine migration in virtualized cluster server based on fuzzy decision making," *arXiv preprint arXiv:1002.3329*, 2010. https://arxiv.org/ftp/arxiv/papers/1002/1002.3329.pdf.

[138] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration.," in *NSDI*, vol. 7, pp. 17–17, 2007.

[139] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pp. 1–15, IEEE, 2011. http://www.informatica.unibo.it/it/ricerca/ technical-report/2011/pdfs/2011-01.pdf.

[140] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble.," *HotPower*, vol. 8, pp. 2–2, 2008.

[141] M. Y. Lim, F. Rawson, T. Bletsch, and V. W. Freeh, "Padd: Power aware domain distribution," in *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pp. 239–247, IEEE, 2009. http://people.engr.ncsu.edu/tkbletsc/pubs/PADD.pdf.

[142] Í. Goiri, J. L. Berral, J. O. Fitó, F. Julià, R. Nou, J. Guitart, R. Gavaldà, and J. Torres, "Energy-efficient and multifaceted resource management for profit-driven virtualized data centers," *Future Generation Computer Systems, Elsevier*, vol. 28, no. 5, pp. 718–731, 2012.

[143] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 243–264, Springer, 2008. https://www.cse.iitb.ac.in/ puru/courses/autumn15/cs695/downloads/pmapper.pdf.

[144] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 conference on USENIX Annual technical conference*, pp. 28–28, USENIX Association, 2009. http://dl.acm.org/citation.cfm?id=1855835.

[145] S. Mehta and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *NOMS 2008-2008 IEEE Network Operations and Management Symposium*, pp. 363–370, IEEE, 2008. http://domino.research.ibm.com/library/ cyberdig.nsf/papers/EA745FA2134A25E38525735A004ABD44 /$File/ri07006.pdf.

[146] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience, Wiley Online Library*, vol. 24, no. 13, pp. 1397–1420, 2012.

[147] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, vol. 4, ACM, 2010.

[148] C. Mastroianni, M. Meo, and G. Papuzzo, "Self-economy in cloud data centers: statistical assignment and migration of virtual machines," in *European Conference on Parallel Processing*, pp. 407–418, Springer, 2011. http://link.springer.com/chapter/ 10.1007%2F978-3-642-23400-2_38.

[149] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 618–624, IEEE, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&ar number=6546147.

[150] H. T. Vu and S. Hwang, "A traffic and power-aware algorithm for virtual machine placement in cloud data center," *International Journal of Grid & Distributed Computing*, vol. 7, no. 1, pp. 350–355, 2014.

[151] J. Sekhar and G. Jeba, "Energy efficient vm live migration in cloud data centers 1," *International Journal of Computer Science and Network (IJCSN), Citeseer*, vol. 2, no. 2, 2013. http://citeseerx.ist.psu.edu/viewdoc/download; jsessionid=ADC017533A8E2166FF5BD488F5AB9A02?doi=10.1.1.300.6376&rep =rep1&type=pdf.

[152] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp. 62–73, IEEE, 2010. https://www.researchgate.net/profile/Gueyoung_Jung /publication/260391823_Mistral_GJung_ICDCS10/links/0f317530fc06ea4e17000000.pdf.

[153] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, "Jettison: efficient idle desktop consolidation with partial vm migration," in *Proceedings of the 7th ACM european conference on Computer Systems*, pp. 211–224, ACM, 2012.

http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/satya/docdir/bila-eurosys 2012.pdf.

[154] P. Graubner, M. Schmidt, and B. Freisleben, "Energy-efficient management of virtual machines in eucalyptus," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 243–250, IEEE, 2011. http://ds.mathematik.uni-marburg.de/system/publications /11873/20110921_050508_Energy-efficient_20Management_20of_20Virtual_20Machines.pdf.

[155] X. Wang and Z. Liu, "An energy-aware vms placement algorithm in cloud computing environment," in *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, pp. 627–630, IEEE, 2012. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber= 6173285.

[156] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review, Taylor & Francis*, vol. 28, no. 3, pp. 212–231, 2011.

[157] L. Minas and B. Ellison, *Energy efficiency for information technology: How to reduce power consumption in servers and data centers.* Intel Press, 2009. http://dl.acm.org/citation.cfm?id=2826078.

[158] F. Dressler and O. B. Akan, "A survey on bio-inspired networking," *Computer Networks, Elsevier*, vol. 54, no. 6, pp. 881–900, 2010.

[159] M. Meisel, V. Pappas, and L. Zhang, "A taxonomy of biologically inspired research in computer networking," *Computer Networks, Elsevier*, vol. 54, no. 6, pp. 901–916, 2010.

[160] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics, IEEE)*, vol. 26, no. 1, pp. 29–41, 1996.

[161] S. A. Ludwig and A. Moallem, "Swarm intelligence approaches for grid load balancing," *Journal of Grid Computing, Springer*, vol. 9, no. 3, pp. 279–301, 2011.

[162] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life reviews, Elsevier*, vol. 2, no. 4, pp. 353–373, 2005.

[163] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005. http://www-lia.deis.unibo.it/Courses/SistInt/articoli /bee-colony1.pdf.

[164] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE control systems, IEEE*, vol. 22, no. 3, pp. 52–67, 2002.

[165] X.-S. Yang, *Nature-inspired metaheuristic algorithms.* Luniver press, 2010. https://books.google.co.in/books?hl=en&lr=&id=iVB_ETlh4ogC&oi=fnd&pg=PR5&dq=Nature-inspired+metaheuristic+algorithms.

[166] F. Xhafa and A. Abraham, "Computational models and heuristic methods for grid scheduling problems," *Future generation computer systems, Elsevier*, vol. 26, no. 4, pp. 608–621, 2010.

[167] W. H. Hsu, "Genetic Algorithms," Tech. Rep. 66506-2302, Department of Computing and Information Sciences, Kansas State University, 234 Nichols Hall, Manhattan, KS, USA.

[168] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[169] M. D. Theys, T. D. Braun, H. Siegal, A. A. Maciejewski, and Y. Kwok, "Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach," *Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences, John Wiley & Sons, New York, NY*, pp. 135–178, 2001. https://www.atmos.colostate.edu/ aam/pdf/books/3.pdf.

[170] A. Abraham, R. Buyya, and B. Nath, "Natures heuristics for scheduling jobs on computational grids," in *The 8th IEEE international conference on advanced computing and communications (ADCOM 2000)*, pp. 45–52, 2000. http://his02.softcomputing.net/adcom.pdf.

[171] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed computing, Elsevier*, vol. 61, no. 6, pp. 810–837, 2001.

[172] T. Kokilavani and D. D. G. Amalarethinam, "Applying non-traditional optimization techniques to task scheduling in grid computing–an overview," *International Journal of Research and Reviews in Computer Science (IJRRCS)*, vol. 1, no. 4, pp. 33–38, 2010.

[173] M. Judy and B. Ramadoss, "An enhanced solution to the protein folding problem using a hybrid genetic algorithm with g-bit improvement strategy," *International Journal of Modeling and Optimization, IACSIT Press*, vol. 2, no. 3, pp. 356–359, 2012.

[174] W. E. Hart, N. Krasnogor, and J. E. Smith, *Recent advances in memetic algorithms*, vol. 166. Springer Science & Business Media, 2004.

[175] A. Colorni, M. Dorigo, V. Maniezzo, *et al.*, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142, pp. 134–142, Paris, France, Springer US, 1991.

[176] P. E. Merloti, "Optimization algorithms inspired by biological ants and swarm behavior," in *SAN DIEGO STATE UNIVERSITY*, no. CS550, Citeseer, 2004.

[177] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence, Springer*, vol. 1, no. 1, pp. 33–57, 2007. http://bee22.com/resources/Poli%202007.pdf.

[178] S. Dasgupta, S. Das, A. Abraham, and A. Biswas, "Adaptive computational chemotaxis in bacterial foraging optimization: an analysis," *IEEE Transactions on Evolutionary Computation, IEEE*, vol. 13, no. 4, pp. 919–941, 2009.

[179] D. Karaboga and B. Basturk, "Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems," in *International Fuzzy Systems Association World Congress*, pp. 789–798, Springer, 2007. https://www.semanticscholar.org/paper/Artificial -Bee-Colony-ABC-Optimization-Algorithm-Karaboga-Basturk/9aecdddc96bc832829 ba35356afa1f10ea0a8930/pdf.

[180] S. Omkar, J. Senthilnath, R. Khandelwal, G. N. Naik, and S. Gopalakrishnan, "Artificial bee colony (abc) for multi-objective design optimization of composite structures," *Applied Soft Computing, Elsevier*, vol. 11, no. 1, pp. 489–499, 2011.

[181] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (abc) algorithm and applications," *Artificial Intelligence Review, Springer*, vol. 42, no. 1, pp. 21–57, 2014.

[182] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International Symposium on Stochastic Algorithms*, pp. 169–178, Springer, 2009. http://arxiv.org/pdf/1003.1466.pdf.

[183] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence, Inderscience Publishers Ltd*, vol. 1, no. 1, pp. 36–50, 2013.

[184] S. R. Khaze, S. Hojjatkhah, A. Bagherinia, *et al.*, "Evaluation the efficiency of artificial bee colony and the firefly algorithm in solving the continuous optimization problem," *arXiv preprint arXiv:1310.7961*, 2013. https://arxiv.org/ftp/arxiv/papers/ 1310/1310.7961.pdf.

[185] B. Basu and G. K. Mahanti, "Fire fly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna," *Progress In Electromagnetics Research B, EMW Publishing*, vol. 32, pp. 169–190, 2011.

[186] C. L. T. Man and M. Kayashima, "Virtual machine placement algorithm for virtualized desktop infrastructure," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 333–337, IEEE, 2011. http://ieeexplore.ieee.org/xpl/login.jsp?tp= &arnumber=6045085.

[187] A. Zhou, S. Wang, Z. Zheng, C.-H. Hsu, M. Lyu, *et al.*, "On cloud service reliability enhancement with optimal resource usage," 2014. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6953216.

[188] A. Esnault, E. Feller, and C. Morin, "Energy-aware distributed ant colony based virtual machine consolidation in iaas clouds bibliographic study," *Informatics Mathematics (INRIA)*, pp. 1–13, 2012. ftp://ftp.irisa.fr/local/caps/DEPOTS/BIBLIO2012/Esnault_Armel.pdf.

[189] S. Voss, "Meta-heuristics : the state of the art," *Local Search for Planning and Scheduling, A. Nareyek (Ed.), Lecture Notes in Artificial Intelligence, Berlin, Springer*, vol. 2148, pp. 1–23, 2001.

[190] N. Bergmann, Y. Y. Chung, X. Yang, Z. Chen, W. C. Yeh, X. He, and R. Jurdak, "Using swarm intelligence to optimize the energy consumption for distributed systems," *Modern Applied Science*, vol. 7, no. 6, pp. 59–66, 2013. DOI:http://dx.doi.org/10.5539/mas.v7n6p59.

[191] X. Wang, Y. Wang, and H. Zhu, "Energy-efficient multi-job scheduling model for cloud computing and its genetic algorithm," *Mathematical Problems in Engineering, Hindawi Publishing Corporation*, 2012. downloads.hindawi.com/journals/mpe/2012/589243.pdf.

[192] F. F. Moghaddam, M. Cheriet, and K. K. Nguyen, "Low carbon virtual private clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 259–266, IEEE, 2011. DOI:http://dx.doi.org/10.1109/CLOUD.2011.36.

[193] N. Quang-Hung, P. D. Nien, N. H. Nam, N. H. Tuong, and N. Thoai, "A genetic algorithm for power-aware virtual machine allocation in private cloud," in *Information and Communication Technology-EurAsia Conference, Lecture Notes in Computer Science*, vol. 7804, pp. 183–191, Springer, 2013. DOI:http://dx.doi.org/10.1007/978-3-642-36818-9_19.

[194] Y. Kessaci, M. Mezmaz, N. Melab, E.-G. Talbi, and D. Tuyttens, "Parallel evolutionary algorithms for energy aware scheduling," in *Intelligent Decision Systems in Large-Scale Distributed Environments*, vol. 362, pp. 75–100, Springer, 2011. DOI:http://dx.doi.org/10.1007/978-3-642-21271-0_4.

[195] L. Chimakurthi *et al.*, "Power efficient resource allocation for clouds using ant colony framework," *arXiv preprint arXiv:1102.2608*, 2011. http://arxiv.org/pdf/1102.2608.pdf.

[196] J. Torres, D. Carrera, K. Hogan, R. Gavaldà, V. Beltran, and N. Poggi, "Reducing wasted resources to help achieve green data centers," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pp. 1–8, IEEE, 2008. https://www.cs.upc.edu/ gavalda/papers/HPPAC08.pdf.

[197] H. Liu, F. Zhong, B. Ouyang, and J. Wu, "An approach for qos-aware web service composition based on improved genetic algorithm," in *Web Information Systems and Mining (WISM), 2010 International Conference on*, vol. 1, pp. 123–128, IEEE, 2010.

[198] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization, Springer*, vol. 39, no. 3, pp. 459–471, 2007.

[199] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience, Wiley Online Library*, vol. 41, no. 1, pp. 23–50, 2011.

[200] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pp. 1–11, IEEE, 2009. https://arxiv.org/ftp/arxiv/papers/0907/0907.4878.pdf.

[201] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 39–50, ACM, 2010. https://www.microsoft.com/en-us/research/wp-content/uploads/2010/06/ JoulemeterVM.pdf.

[202] L. Hongyou, W. Jiangyong, P. Jian, W. Junfeng, and L. Tang, "Energy-aware scheduling scheme using workload-aware consolidation technique in cloud data centres," *China Communications, IEEE*, vol. 10, no. 12, pp. 114–124, 2013.

[203] C. Jo, E. Gustafsson, J. Son, and B. Egger, "Efficient live migration of virtual machines using shared storage," in *ACM Sigplan Notices*, vol. 48, pp. 41–50, ACM, 2013.

[204] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pp. 514–521, IEEE, 2013. http://www.rn.inf.tu-dresden.de/uploads /Publikationen/aina2013.pdf.

[205] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, "Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing," *Computing, Springer*, vol. 98, no. 3, pp. 303–317, 2016.

[206] S. Jain and V. Sharma, "Enhanced load balancing approach to optimize the performance of the cloud service using virtual machine migration," *International Journal of Engineering and Manufacturing(IJEM)*, vol. 7, no. 1, pp. 41–48, 2017. 10.5815/ijem.2017.01.04.

[207] Y. Gao and L. Yu, "Energy-aware load balancing in heterogeneous cloud data centers," in *Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences*, pp. 80–84, ACM, 2017.

[208] M. Vanitha and P. Marikkannu, "Effective resource utilization in cloud environment through a dynamic well-organized load balancing algorithm for virtual machines," *Computers & Electrical Engineering, Elsevier*, vol. 57, pp. 199–208, 2017.

[209] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-mmt: A load balancing method for power consumption management in cloud computing," in *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pp. 76–80, IEEE, 2013.

[210] S. Bose and J. Kumar, "An energy aware cloud load balancing technique using dynamic placement of virtualized resources," *Advances in Computer Science and Information Technology (ACSIT)*, vol. 2, no. 7, pp. 81–86, 2015.

[211] A. Goyal and N. S. Chahal, "Bio inspired approach for load balancing to reduce energy consumption in cloud data center," in *Communication, Control and Intelligent Systems (CCIS), 2015*, pp. 406–410, IEEE, 2015.

[212] V. R. Rajyashree, "Double threshold based load balancing approach by using vm migration for the cloud computing environment," *Int J Eng Comput Sci*, vol. 4, no. 1, pp. 9966–9970, 2015.

List of Publications

International Journals (indexed by SCI)

1. Nidhi Jain Kansal and Inderveer Chana, "Artificial bee colony based energy-aware resource utilization technique for cloud computing", Concurrency and Computation : Practice and Experience (CCPE), Vol. 27, Issue 5, pp. 1207-1225, May 2014, doi:10.1002/cpe.3295, *(**Published** by Wiley Online Library, Impact Factor: 0.997, ISSN: 1532-0634)*, **cited** by 9.

2. Nidhi Jain Kansal and Inderveer Chana, "Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach", Journal of Grid Computing (JoGC), Vol. 14, Issue 2, pp. 327-345, February 2016, doi:10.1007/s10723-016-9364-0, *(**Published** by Springer, Impact Factor: 1.507, ISSN: 1570-7873)*, **cited** by 2.

International Journals (not indexed by SCI)

1. Nidhi Jain Kansal and Inderveer Chana, "Cloud Load Balancing Techniques: A Step Towards Green Computing", International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 1, No 1, pp. 238-246, January 2012, *(**Published** by IJCSI, Impact Factor: 0.242 , ISSN: 1694-0814)*, **cited** by 127.

2. Nidhi Jain Kansal and Inderveer Chana, "Existing Load Balancing Techniques In Cloud Computing: A Systematic Review", Journal of Information Systems and Communication (JISC), Vol. 3, Issue 1, pp. 87-91, March 2012, doi: 10.9735/0976-8742, *(**Published** by Bioinfo Publications, ISSN: 0976-8742)*, **cited** by 46.

Paper Communicated

1. Nidhi Jain Kansal and Inderveer Chana, "An Empirical Evaluation of Energy-aware Load Balancing Technique for Cloud Computing", Journal of Cluster Computing (JoCC), *(**Communicated**, Springer, Impact Factor: 1.510, ISSN: 1386-7857)*.