Energy-Efficient Resource Allocation in Green Clouds

A Thesis

submitted in partial fulfillment of the requirements for the degree of

Doctorate of Philosophy

by

Ashok Kumar

(Reg. No.:951303004)

under the supervision of

Dr. Rajesh Kumar
Professor
Computer Science & Engineering Department
Thapar University
Patiala, Punjab

Dr. Anju Sharma
Assistant Professor
Computer Application Department
MRS Punjab Technical University
Bathinda, Punjab

Computer Science and Engineering Department
Thapar University
Patiala-147004, Punjab, India
January, 2017

Certificate

I hereby certify that the work which is being presented in this thesis titled, **"Energy-Efficient Resource Allocation in Green Clouds"**, in partial 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. Rajesh Kumar and Dr. Anju Sharma, and refers other research 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.

(Ashok Kumar)

Reg. No. 951303004

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

20/11/2017

Dr. Rajesh Kumar
Professor
Computer Science & Engineering Department
Thapar University
Patiala, Punjab

Dr. Anju Sharma
Assistant Professor
Computer Application Department
MRS Punjab Technical University
Bathinda, Punjab

Acknowledgements

PhD is a once in a lifetime opportunity. It is an exhausting, and emotional struggling which demands regress hard work and discipline. I am in ecstasy that, with the grace of almighty God, I have had a chance to complete it. It would not be possible without the support of many people. I would like to take this opportunity to thank the people who inspired me during the ups and downs of this pleasant experience. First and foremost, I would like to express my sincere gratitude towards my PhD supervisors, Dr. Rajesh Kumar, Professor, Computer Science and Engineering Department, Thapar University and Dr. Anju Sharma, Assistant Professor, Maharaja Ranjit Singh Punjab Technical University, Bathinda for giving me this opportunity. Both of them gave me guidance, support, and encouragement throughout my PhD candidature. Their contribution to this thesis goes well beyond the role of an academic supervisor. They helped me even when I encountered personal issues, and supported me as a family. I am really indebted to them for providing me motivating and enthusiastic atmosphere to work in. I would like to express my gratitude to Dr. Maninder Singh, Head, Computer Science and Engineering Department, and Dr. O. P. Pandey, Senior Professor & Dean-RSP, for their constant motivation and encouragement. I would like to express my gratitude towards the doctoral committee members, Professor R. K. Sharma, Professor Inderveer Chana, and Dr. Rajiv Kumar for their constructive comments and invaluable suggestions. Many thanks to all my friends Inderpreet Singh, Sukhveer Singh, Sandeep Singh, Gourav Gupta, Tarun Kumar, Santosh Kumar, Sharad Tiwari, Harjeet Singh, Gurjinder Singh, and Simar Preet Singh to name a few for cooperation and help they extended as and when required. I would like to give thanks to my parents, parents-in-law, brothers, brothers-in-law, sisters and sisters-in-law for their constant support, encouraging words, love, and support. I want to thank Dr. Jagpreet Singh, my fast friend, who encouraged me to pursue PhD. I would also like to thank all those who supported me at any stage in my work, from the inception to the completion of this thesis. I am also very thankful to the entire faculty and staff members of Computer Science and Engineering Department, Thapar University, Patiala for their direct and indirect help, cooperation, love and affection. I dedicate this PhD thesis to my lovely children, Aarzoo and Pratham who are the pride and joy of my life.

Lastly and most importantly, I would also like to dedicate this thesis to my beloved wife, Yogita Khunger, who has been making my life so incredible and prosperous each and every day. These few words cannot express my deepest appreciation for her selfless patience, unconditional love, and endless support during these past years. I cordially thank you for being so understanding and putting up with me through the toughest moments of my life. I thank God for enlightening my life with your presence.

Ashok Kumar

Abstract

Cloud Computing is a business model that is being widely-adopted by enterprises and organizations. It has become the preferred computing platform for deploying applications and services owing to its inherent characteristics, including scalability, pay-per-use, rapid elasticity, cost saving, self-service, and broad network access. These characteristics of Cloud computing have played a major role in its widespread adoption. To accommodate the growing demand for computational resources, Cloud market players Amazon, Microsoft, Google, GoGrid, Flexiant, etc. have set up large sized Data Centers (DCs). These large scale DCs consume huge amount of energy. Further, with the proliferation of Cloud computing, more and more Cloud based applications with varying resource demands and diverse Quality of Service (QoS) requirements are coming up, which require dynamic allocation, reconfiguration, and reallocation of resources. All these requirements necessitate the development of energy-efficient and QoS aware resource allocation techniques that not only reduce energy consumption but also satisfy QoS requirements of the end users. Reducing energy consumption while providing QoS is the biggest challenge that the Cloud service providers are confronting with.
Therefore, to achieve the set objectives of energy efficient resource allocation, an extensive literature review on resource allocation in Cloud computing has been done. The state of the art techniques in the area of power management and resource allocation in Clouds have been explored. The comprehensive study of energy-efficient resource allocation techniques in Clouds has been carried out to identify their inherent limitations. From the literature survey, it is apparent that the biggest challenge confronting Cloud service providers is related to energy consumption and QoS aware resource allocation.

To address the energy and QoS related challenges of Cloud, a green Cloud framework, named "SERVmegh", has been proposed for efficient and robust management of resources. The framework considers various characteristics of the Cloud such as robustness, scalability, fault-tolerance, and energy efficiency. The proposed framework divides the entire functionality across the different layers and also provides well defined interfaces for communication between layers. The layered architecture bears negligible overhead in terms of communication cost. A comparative analysis of the

proposed framework with various open source as well commercial Cloud frameworks has been done. Further, a resource wastage reduction based allocation technique has been proposed for reducing energy consumption. The performance analysis of the proposed technique has been carried out in simulated as well as OpenNebula based private Cloud environment. The comparative analysis of the proposed technique with state of the art resource allocation technique has shown up to 18% reduction in energy consumption.

To enhance the utility of "SERVmegh" cloud, a resource allocation technique based on Ant Colony Optimization (ACO) has been proposed and implemented. The resources have been allocated to the jobs to minimize total cost of execution, total execution time and total energy consumption while satisfying QoS requirements of the end users. Each QoS parameter of a job has been assigned some weight value that indicates its priority over the others. ACO has been applied at two levels for efficient allocation of resources. First level ACO allocates jobs to Virtual Machines (VMs), whereas second level ACO allocates VMs to Physical Machines (PMs). Server consolidation and dynamic performance scaling has been employed to conserve energy. The effectiveness of this approach is evaluated in CloudSim with jobs having different resource demands and QoS requirements. Approximately 10% improvement in energy consumption has been observed from comparative analysis with existing resource allocation techniques.

Further, an improved energy and QoS aware resource allocation technique "EQUAL" has also been proposed and implemented. EQUAL is based on Antlion optimization for efficient allocation of resources to an application encapsulated in a Virtual Machine(VM). It can be operated in three modes, namely power aware mode, performance aware mode, and balanced mode. When operated in power aware mode, it strives minimize number of PMs in order to reduce energy consumption. In performance mode, the applications has been allocated to servers that have maximum available capacity at disposal. Whereas, in balanced mode, power and performance have been given equal weightage while allocating resources to the applications. The mode of operation can be changed by varying the values of the control variables. The proposed approach is implemented in CloudSim simulator for performance evaluation

and validation. The experimental results show that EQUAL saves energy upto 15% and improves QoS in terms of reduction in percentage of tasks missed their deadlines.

The major issue with EQUAL is, that it can not switch between different modes automatically. The mode of operation has to be selected offline. This issue has been addressed by extending it to support self-optimization of resources. The proposed self-optimizing resource allocation system can automatically switch to power mode, performance mode, or balanced mode in accordance with the workload of the application. The necessary changes are made in the control variables to shift the resource allocation system to different mode of operation. For autonomic mode switching, resource utilization history of the applications is used for predicting their resource needs in the near future. The self-optimization of resources allocated to an applications has been carried out in four phases: *Monitor*, *Analyze*, *Plan* and *Execute*. The VMs are migrated to offload heavily loaded machines or consolidate lightly loaded machines to fewer physical machines to save energy by switching idle machines to low power modes. Server selection for a VM has been carried out using Antlion optimization. The self-optimizing resource allocation system has been evaluated with CloudSim simulator. The results have shown improvement in energy-efficiency and QoS of the Clouds. The results show that the proposed energy-efficient and QoS-aware resource allocation technique efficaciously addresses the challenges of cloud.

List of Publications

Refereed Journals: Published/Accepted

1. **Ashok Kumar**, Anju Sharma, Rajesh Kumar "SERVmegh: framework for green cloud", Concurrency and Computation: Practice and Experience, Wiley-SCI Indexed, ISSN 1532-0634, 2016, [I.F. 0.942].

2. **Ashok Kumar**, Rajesh Kumar, Anju Sharma "Energy Aware Resource Allocation for Clouds Using Two Level Ant Colony Optimization", Computing & Informatics-SCI Indexed [accepted] , ISSN 1335-9150, 2016, [I.F. 0.524].

3. **Ashok Kumar**, Rajesh Kumar, Anju Sharma "EQUAL: Energy and QoS Aware Resource Allocation Approach for Clouds", Computing & Informatics-SCI Indexed [accepted], ISSN 1335-9150, 2016, [I.F. 0.524].

Refereed Journals: Communicated

1. **Ashok Kumar**, Rajesh Kumar, Anju Sharma "A Self Optimizing System for Energy and Quality of Service Aware Resource Allocation in Clouds", Computers & Electrical Engineering, , ELSEVIER-SCI Indexed, ISSN 0045-7906, 2016, [I.F. 1.084].

2. **Ashok Kumar**, Anju Sharma, Rajesh Kumar "State of the art Energy Efficient Techniques for Resource Allocation in Green Clouds: A Survey", Concurrency and Computation: Practice and Experience, Wiley-SCI Indexed, ISSN 1532-0634, [I.F. 0.942].

Conferences

1. Rajesh Kumar, **Ashok Kumar**, Anju Sharma, "A Bio-inspired Approach for Power and Performance Aware Resource Allocation in Clouds", in: 4th International Conference on Advancements in Engineering and Technology (ICAET-2016), ISSN 1532-0634, 2016

Contents

List of Figures

List of Tables

List of Notations

Chapter 3

δ	Set of states in DTMC
M	Number of physical machines or servers
$s_0, s_1, ..., s_{n+1}$	DTMC states
N	Number of virtual machines
P	Transition probability matrix
S	Set of server/physical machines
V_i	Number of visits to state s_i
$C_i^{ps}, C_i^{mem}, C_i^{nbw}$	Processing speed, memory, and network bandwidth of ith server
q_i	Probability of starting service request at state s_i
$U_i^{ps}, U_i^{mem}, U_i^{nbw}$	Processing speed, memory, and bandwidth utilization of ith server
TR^p	Total CPU requirements of all layers
$r_j^{ps}, r_j^{mem}, r_j^{nbw}$	Processing speed, memory, and bandwidth requirements of jth VM
TR^d	Total disk requirements of all layers
$R_i^{ps}, R_i^{mem}, R_i^{nbw}$	Unused processing speed, memory, and bandwidth of ith server
TR^c	Total service requirements of all connectors
x	$M.N$ array for VMs to servers mapping
CPU_i	Average CPU time per service request
ω_i	Normalized resource wastage vector of ith server
$Disk_i$	Average disk time per service request
$\Re_i$	Wastage reduction factor of ith server
PM_L	Set of PMs on which all layers are deployed
y_i	is 1 if ith PM is in use, 0 otherwise
L_j	Set of layers collocated on server j
T_w^{lower}	Lower wastage threshold
T^p	Total processing cost
T_w^{upper}	Upper wastage threshold
T^c	Total cost of communication

C_i	Capacity vector of i^{th} PM
T^o	Overhead cost
r_j	Request vector of j^{th} VM
ζ	Unit cost of CPU
U_i	Utilization vector of i^{th} PM
ξ	Unit cost of disk
R_i	Unused resource capacity vector of i^{th} PM
γ	Unit cost of communication
$packet_i$	Size of packet transmitted by connector i
κ	Set of all connectors
cap_i	Communication capacity of i^{th} connector

Chapter 4

CL	Set of clusters
cl_i	ith cluster
PM_i	Set of physical machines in ith cluster
N_c	Number of clusters
N_v	Number of VMs
N_t	Number of tasks/jobs
N_a^1	Number of ants used in allocating jobs to VMs
N_a^2	Number of ants used in allocating VMs to PMs
H_i	Number of PMs in cluster i
h_{ij}	j^{th} PM of cluster i
$hs_{ij}, hm_{ij}, hn_{ij}$	CPU speed, memory, and network bandwidth of j^{th} PM of cluster i
E_{ijg}	Energy consumed by g^{th} VM if executed on j^{th} PM of cluster i
E_{ij}	Energy consumption of j^{th} PM of cluster i
VM	Set of VMs
R_g	Requirement of g^{th} VM
r_{ij}^q	Resource requirement of task q executing on j^{th} PM of cluster i
t_{ij}^q	Execution time of task q on j^{th} PM of cluster i
p_{ij}^q	Power consumed for executing task q on j^{th} PM of cluster i

s_{ij}^{q}	CPU speed allocated to task q on j^{th} PM of cluster i
e_{ij}^{q}	Energy consumed for executing task q on j^{th} PM of Cluster i
T_{ijg}	Total execution time of all tasks deployed on g^{th} VM
h	Number of voltage/frequency levels supported by PM
C_{ijg}	Total cost of running g^{th} VM on j^{th} PM of cluster i
x_{ijg}	1 if g^{th} VM is assigned to j^{th} PM of cluster i, 0 otherwise
EC_{ij}	Execution cost of VMs run on j^{th} PM of cluster i
v_g	g^{th} VM
vs_g, vm_g, vn_g	CPU speed, memory, and network bandwidth of g^{th} VM
n_g	Number of jobs allocated to g^{th} VM
G_1	Construction graph for jobs to VMs mapping
N_1	Set of nodes of construction graph G_1
L_1	Set of edges of construction graph G_1
α_1	Parameter to control influence of pheromone trail in G_1
β_1	Parameter to control influence of heuristic information in G_1
ρ_1	Pheromone evaporation rate for graph G_1
$\wp_{ag}^{k}$	Probability of mapping job a to g^{th} VM
W_{ax}	Weight value of QoS parameter x of job a
ℓ_a	Length of job a
X	Number of QoS parameters
y_{ag}	is 1 if job a is assigned to g^{th} VM, 0 otherwise
$\mathcal{N}_g^k$	Feasible neighborhood of VM g for ant k
$S1^k$	Ant k's jobs to VMs mapping solution
$D1^k$	Number of VMs used in solution k
G_2	Construction graph for VMs to PMs mapping
N_2	Set of nodes of construction graph G_2
L_2	Set of edges of construction graph G_2
$\tau_{u,v}$	Pheromone value associated with edge (u, v)
$\eta_{u,v}$	Heuristic information associated with edge (u, v)
α_2	Parameter to control influence of pheromone trail in G_2
β_2	Parameter to control influence of heuristic information in G_2
ρ_2	Pheromone evaporation rate for graph G_2

$\mathcal{N}_j^k$	Feasible neighborhood of PM j for ant k
FM_j	Available memory space of j^{th} PM
$S2^k$	Ant k's VMs to PMs mapping solution
$D2^k$	Length of solution k
d_{ij}	Distance of j^{th} PM of cluster i from frontend server
i, j, g, k, q, k	Identifier for cluster, PM, VM, ant, task, and ant, respectively
ξ, ζ, γ	Weight values for TEC, TET, and COE, respectively
P_{idle}	Power consumption of idle PM
P_{max}	Power consumption of PM at 100% utilization
U	Utilization of a PM
P	Power consumption of PM at U% utilization
CPU_{UGT}	Upper Green Threshold value for CPU
MEM_{UGT}	Upper Green Threshold value for Memory
BW_{UGT}	Upper Green Threshold value for Bandwidth
CPU_{LGT}	Lower Green Threshold value for CPU
MEM_{LGT}	Lower Green Threshold value for Memory
BW_{LGT}	Lower Green Threshold value for Bandwidth
$P_{dynamic}$	Dynamic power consumption
P_{static}	Static power consumption
A	Switching activity
C	Capacitance
V	Voltage
I_{leak}	Leaking current
f	Frequency

Chapter 5

P_{total}	Total power consumption of a physical machine
$P_{dynamic}$	Dynamic power consumption of a physical machine
P_{static}	Static power consumption of a physical machine
P_{idle}	Power consumption of a physical machine when idle
P_{max}	Power consumption of PM at 100% utilization

U	Utilization of a PM
P	Power consumption of PM at $U\%$ utilization
e	Elite solution representing the best resource
n	Number of tasks
J_i	i th task
d_i	Deadline of task J_i
w_i	Processing volume of task J_i
V	Number of virtual machines
m	Number of resources
S	Set of resources
A	Set of ants
L	Set of antlions
S_j	j th resource
$\Re_r^a$	Available processing power of resource r
$\Re_j^d$	Processing demand of VM j
$\triangle E_{j,r}$	Energy contribution of VM j on resource r
κ_r	Energy affinity
$\Re_{i,r}$	Fraction of processing power allocated to VM i on resource r
γ	Constant that controls energy contribution and VMs consolidation
θ	Trade off between performance and energy
$f_{j,r}$	Fitness of VM j on resource r
M^a	Matrix to store location of ants
M^{al}	Matrix to store location of antlions
M^{fa}	Matrix to store fitness values of ants
M^{fal}	Matrix to store fitness values of antlions
W_i^t	Location of i^{th} ant at t^{th} iteration
V_j^t	Location of j^{th} antlion at t^{th} iteration
α	Scaling factor for step size s
λ	Levy exponent
$L(s, \lambda)$	Levy Distribution with parameters s and λ
a_i	Minimum of random walk of i^{th} ant
b_i	Maximum of random walk of i^{th} ant

c_i^t	Minimum of search space at t^{th} iteration
d_i^t	Maximum of search space at t^{th} iteration
M_{VR}	VM-Resource map
UGT	Upper Green Threshold
LGT	Lower Green Threshold
HT	Hot-spot Threshold
CT	Cold-spot Threshold

Chapter 6

M_{VR}	VM-Resource map
u_i	Utilization of i^{th} application
$< T_l$	Lower threshold limit
θ_j	Control Variable
T_u	Upper threshold limit
$f_{j,r}$	Fitness of resource r for application j
$\Re_r^a$	Available processing power of resource r
$\Re_j^d$	Processing demand of j^{th} application
$\triangle E_{j,r}$	Energy contribution of application j on resource r
κ_r	Energy affinity
$\Re_{i,r}$	Fraction of processing power allocated to application i on resource r

List of Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ACPI	Advanced Configuration and Power Interface
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AWS	Amazon Web Services
BFD	Best Fit Decreasing
CAPEX	Capital Expenditure
CED	Carbon Emission Directory
CLI	Command Line Interface
COE	Total Cost of Execution
CRM	Customer Relationship Management
DC	Data Center
DPC	Dynamic Power Consumption
DPM	Dynamic Power Management
DTMC	Discrete Time Markov Chain
DVFS	Dynamic Voltage Frequency Scaling
EARA	Energy Aware Resource Allocation
EC2	Amazon Elastic Compute Cloud
EDF	Earliest Deadline First
EQUAL	Energy and QoS aware resource allocation approach Using AntLion optimization
FCFS	First Come First Serve
FFD	First Fit Decreasing
GA	Genetic Algorithm
GHA	Geometric Heuristic Algorithm
GIC	Global Information Collector
GNC	Global Node Controller
GOD	Green Offer Directory
GUI	Graphical User Interface

HPC	High-performance computing
I/O	Input/Output
IaaS	Infrastructure as a Service
ICT	Information Communication and Technology
IP	Information Probes
IQR	Interquartile Range
iSCSI	Internet Small Computer System Interface
IT	Information Technology
KVM	Kernal-based Virtual Machine
LF	Levy Flight
LGT	Lower Green Threshold
LNC	Local Node Controller
LUDB	Local Utilization Database
LVM	Logical Volume Management
MAC	Media Access Control
MAPE	Monitor-Analyze-Plan-Execute
MB	Megabyte
MEM	Memory
MGGA	Multi-objective Grouping Genetic Algorithm
MI	Millions of Instructions
MIPS	Millions of Instructions Per Second
MMST	Multi-Model Switching and Tuning
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NBW	Network Bandwidth
NFS	Network File System
NQRA	Non-QoS aware Resource Allocation
NRWV	Normalized Resource Wastage Vector
ON	Open Nebula
OPEX	Operational Expenditure
OS	Operating System
PaaS	Platform as a Service

PCs	Personal Computers
PDA	Personal Digital Assistant
PM	Physical Machine
PS	Processing Speed
QoS	Quality of Service
RA	Resource Allocation
RBMA	Reverse Batch Matching Auction
REST	Representational State Transfer
ROI	Return on Investment
RS	Resource Scheduler
RU	Resource Utilization
RWR	Resource Wastage Reduction
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOCCER	Self-Optimization of Energy-efficient Cloud resources
SOS	Self-Optimizing System
SPC	Static Power Consumption
SPM	Static Power Management
SSH	Secure Shell
SVRs	Support Vector Regressions
TCO	Total Cost of Ownership
TEC	Total Energy Consumption
TET	Total Execution Time
TGT	Ticket-Granting Ticket
UDP	User Datagram Protocol
UGT	Upper Green Threshold
UIDB	Utilization Information Database
VBP	Vector Bin Packing
VM	Virtual Machine
VMFS	Virtual Machine File System
VMM	Virtual Machine Monitor

VNC	Virtual Network Computing
VO	Virtual Organization
WA	Workload Analyzer
WLDB	Workload Database
WRF	Wastage Reduction Factor
WRM	Wastage Reduction Manager
XaaS	Anything as a Service

Chapter 1

Introduction

Cloud computing is the result of evolution and adoption of multiple computing paradigms, including virtualization, distributed computing, grid computing, and utility computing. It has emerged as a propelling computing paradigm for delivering services over the internet on pay-per-use basis. With its use, the consumers can cut costs and focus on their main business activities instead of being hindered by Information Technology (IT) obstacles. The term Cloud come into being dates back to the 1960s when John McCarthy put forth the thought of delivering computing as a public utility, just like gasoline, water and electricity. However, the giant step towards Cloud computing was taken in the mid 1990s when grid computing evolved to deliver computing capabilities on demand. It emerged from grid computing as service model and augments grid services with technologies like virtualization and a business model.

The chapter starts with overview of Cloud computing, and then describes how it evolves from the various key computing technologies. Further, it presents the essential characteristics, various types of services and deployment models supported by Cloud computing paradigm. The chapter also introduces Green Cloud computing, and some of the key foundation technologies from which Cloud computing has evolved. Further, it elucidates Cloud adoption challenges, motivation to energy efficient resource allocation, and organization of the thesis. The chapter concludes with thesis contributions.

1.1 Cloud Computing Overview

Cloud computing is a new business and service model for delivering services over the internet or intranet on pay-per-usage basis [1, 2]. The evolution paths of many computing paradigms, including service-oriented architecture, grid computing, utility

computing, autonomic computing, and virtualization meet at the Cloud computing juncture. Thus, it provides the benefits of all these computing paradigms in addition to the benefits that are specific to Cloud computing. It supports real time development, deployment and delivery of a wide range of services which can be accessed through a variety of devices, including personal computers, laptops, smart-phones, and Personal Digital Assistant (PDA) via thin client (web browser).

Although, Cloud computing has attracted significant momentum and attention in both academia and industry, yet there is no consensus on a definition of Cloud computing. All the proposed definitions of Cloud computing revolve around the characteristics that it offers, including cost savings, high availability, and easy scalability. The most comprehensive definition of a Cloud is: *"A Cloud is a type of parallel and distributed computing system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumer"* [3].

The term Cloud is considered as metaphor of internet based services. The six phases, adapted from [4], through which Cloud computing evolved from mainframes is as shown in Figure 1.1. In phase 1, huge computing power of a mainframe computer was shared by multiple users through dumb terminals. Then, in phase 2, the Personal Computers (PCs) became powerful enough to fulfill the computational demands of the users, so the people started using their individual PCs for their computational needs. With the passage of time the computational needs of users' exceeded beyond the capacity of PCs. Therefore, to fulfill the computational requirements in phase 3, they interconnected their PCs to form a network in order to leverage computational power of many PCs. The invention of internet, in phase 4, made it possible to interconnect geographically dispersed local networks to form a global network which enabled remote access to applications and resources through the internet. In phase 5, grid computing evolved which provided shared computing power and storage via a distributed computing system. In phase 6, Cloud computing came into being and provided access to pool of computing resources through PC over the Internet in a scalable and simple manner. Cloud computing is considered to have returned to the mainframe computing, where the dumb terminals were used to share the huge computing power. But

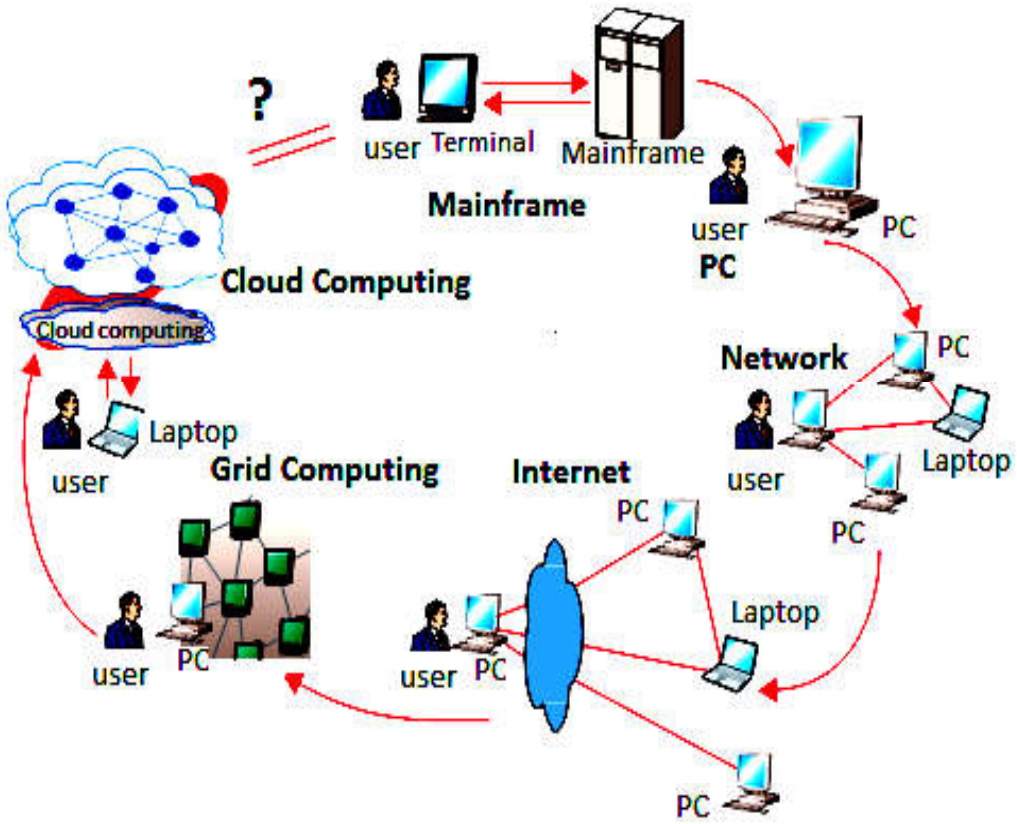


Figure 1.1: Six phases of evolution from mainframe to Cloud computing [4]

significant difference exists between the two paradigms in terms of computing power and capacity. Cloud computing allows the users to exploit all the available computing resources through the internet and hence considered to have infinite computing power. However, a mainframe computer has a limited computing power. Similarly, a dumb terminal, a little more than a keyboard and a monitor, was used as an interface in mainframes. In contrast, a PC in the Cloud computing possesses significant computing power and thus provides a certain degree of local computing and caching support. Thus, Cloud computing has emerged due to the flourishing of the internet and information technology industry and completes the transformation from the traditional, single-task-oriented computing model to a modern, service-oriented, multi-computing model. Further, Cloud computing enables the Cloud consumers to utilize the services provided by the Cloud service providers in a location independent way. Cloud services are hosted on the service providers' own infrastructure or on a third party Cloud infrastructure. The service providers earn profit by charging consumers for the type

and duration of access of the services. The Cloud consumers can rent the services from Cloud providers, eliminating the need of making huge capital investments, thus shifting the cost of business from Capital Expenditure (CAPEX) to Operational Expenditure (OPEX). Cloud computing offers immense flexibility and affordability to help the development of application with ease, and carry out the business activities with immediate scaling and cost effective manner.

After an overview of Cloud computing, the next section discusses evolution and emergence of Cloud computing from cluster and grid computing paradigms.

1.1.1 Cloud Computing Evolution

Cloud computing has evolved from various computing paradigms over the years. The Figure 1.2 shows the major technology milestones that have led to Cloud computing. The progressive change in technology over the years has caused contextual overlap of these computing paradigms.

A cluster is a collection of interconnected computers that are managed to provide

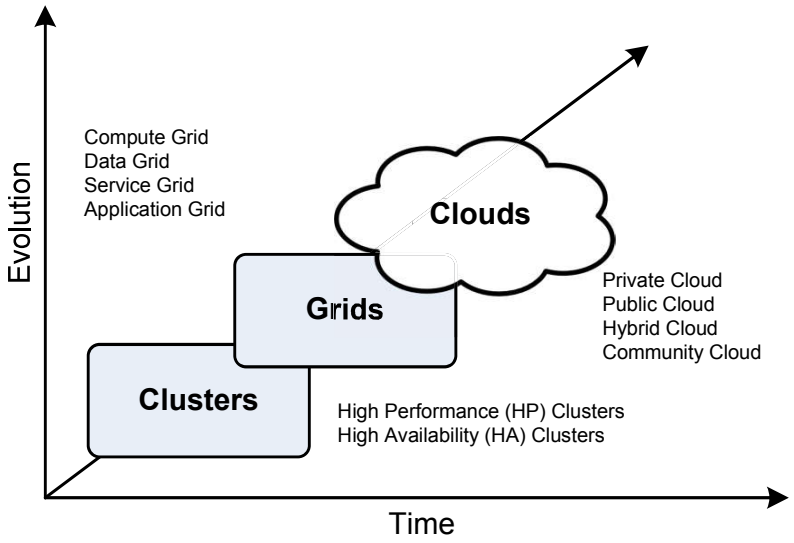


Figure 1.2: Evolution of distributed computing paradigm

the computing power equivalent to a large supercomputer with single administrative domain, thereby enhancing the overall performance and availability [5].

With the increase in computing requirements over time, different organizations come together to achieve a common goal through pooling their computing infrastructure.

The infrastructure of different organizations that is connected through the internet to provide a huge computational power is termed as Grid. The grids provide massive computing power for complex applications and therefore referred to as computing infrastructure for e-science [6]. In computational grid, a complex job is divided into many subtasks which are distributed to multiple machines for execution. Each computing node is set to perform the same task that is controlled and scheduled by software. Unlike cluster computing, in grid computing each node is set to perform a different task. Grid computing provides cost effective services, reliability, availability, and deliver computing power on demand [7]. Unlike Grids, Cloud provides computing as a utility to consumers.

The various characteristics delivered by Cloud computing are provided through Cloud service models which are discussed in the next section.

1.1.2 Cloud Architecture

Cloud architecture can typically be modeled as a set of three layers consisting of underlying hardware, platform and application software. The different components of these three layers are delivered as services with different level of control to the Cloud consumers. The services provided through different layers are accordingly called Infrastructure as a Service, Platform as as Service, and Software as a Service as shown in Figure 1.3 [1]. During the past decade, Cloud computing had progressed enough to to deliver "Anything" as a Service (XaaS) to the users, where X means any arbitrary service. For instance, Cloud computing now-a-days offers variety of services including network as a service, storage as a service, desktop as a service, and database as a service which are considered as integral part of the cloud service model.

The following subsection give a brief overview of these service models.

1.1.2.1 Service Models

Infrastructure as a Service (IaaS): This model delivers infrastructure resources such as server, networking, storage, firewall, and load balancer on rental basis. The users

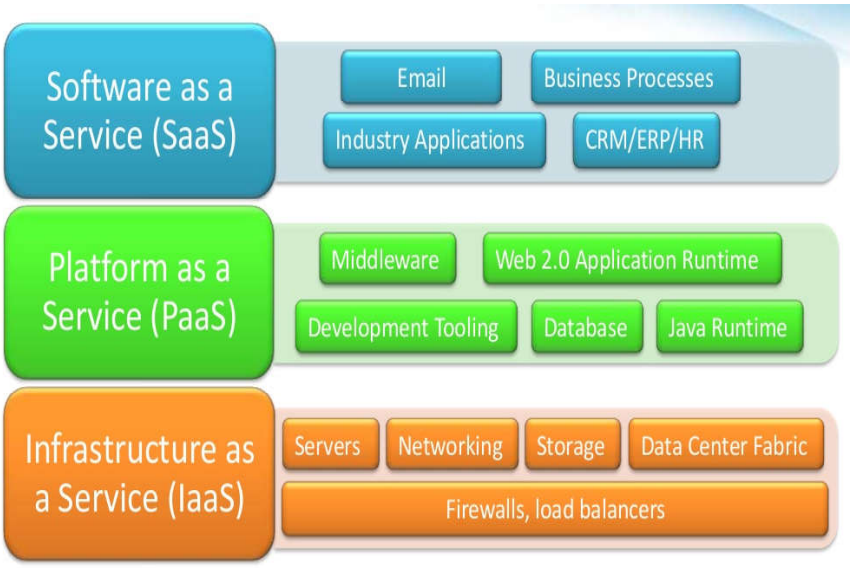


Figure 1.3: Cloud service layer architecture

can deploy and run arbitrary application, operating system and software on the provisioned resources. The consumers have the advantage of provisioning resources on rent rather than spending money for buying their own servers and networking equipment. Some of the commercial IaaS providers are Amazon Elastic Compute Cloud (EC2) [8], Rackspace [9], and FlexiScale [10].

Platform as a Service (PaaS): PaaS model provides high-level integrated environment like database, Web 2.0 application runtime, and development tools for hassle free application development onto the Cloud platform. The customers can manage the deployed applications and their hosting environment configurations, but do not have control over the underlying hosting environment. The leading PaaS providers are Google AppEngine [11], Microsoft Azure [12] and Heroku [13].

Software as a Service (SaaS): It provides users access to applications such as email, CRM/ERP/HR and business processes deployed on the Cloud through a web browser. Further, single instance of an application can serve multiple users or client organizations. Applications such as Customer Relationship Management (CRM), Live Mesh, and Google Docs enrich the family of SaaS-based services which are offered by Salesforce.com [14], Microsoft [15], and Google [16].

The service model of Cloud computing is innovative in nature. Despite many advantages, Cloud computing brings some challenges with it, such as security, reliability and elasticity, which immensely affect the Cloud users. In order to solve these serious issues, industry divides the application deployment into four models on the basis of the relationship between infrastructure providers and service consumers as shown in Figure 1.4. These four deployment models are briefly discussed as follows:

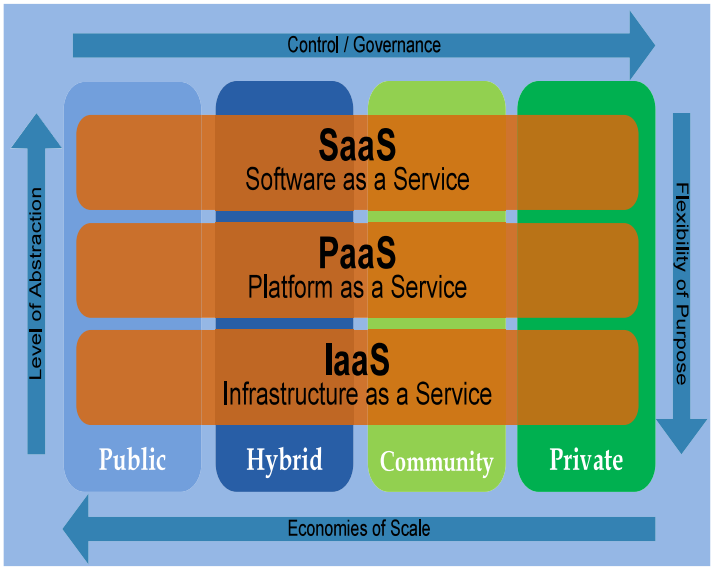


Figure 1.4: Cloud computing deployment models

1.1.2.2 Deployment Models

Private Cloud: Private Cloud is implemented on the computational infrastructure owned by the organization. The users of the organization are the clients of the Cloud. The operational and maintenance costs of the Cloud are borne by the organization. Private Cloud provides high security for data that comes at the cost of high initial investment. Further, highly skilled labour is required for the implementation of the Cloud. Private Cloud can be setup using open source Cloud tools including OpenNebula [17], and OpenStack [18].

Public Cloud: Public Cloud infrastructure is owned, managed, and maintained by the third party selling Cloud services, and therefore reduces customers' risk of investing in the infrastructure. Public clouds offer services to the general public on a pay-per-use

basis. Examples of public clouds include Amazon Elastic Compute Cloud (EC2) [8], and Oracle Cloud [19]

Hybrid Cloud: The usage of public Cloud in conjunction with private Cloud is called hybrid cloud. Hybrid clouds provide benefits of both public and private clouds at the expense of additional management system for coordination between the Clouds. An organization utilizing hybrid Cloud, manages private Cloud in-house for its normal usage, and hires the additional resources from public cloud providers on demand to accommodate load spike. For example, an organization might store customer data within its own data center for privacy and may hire public Cloud, such as Amazon's EC2, to provide computing power on-demand when data processing is needed.

Community Cloud: Community Cloud is utilized by specific community of consumers from different organizations having a common mission, security policy, or compliance concerns. It is managed by some of the organizations from the community.

Clouds support certain key characteristics as listed in the subsection below.

1.1.3 Essential Characteristics

Cloud computing has its roots deep in the distributed computing, and therefore provides the benefits it shares with distributed computing paradigms inclusive of adaptability, scalability, resource sharing, dynamism and security. The characteristics that distinguish Clouds from other computing paradigms are listed below [1].

- *On-demand Self-service*: The consumers can themselves provision the computing capabilities such as server time and network storage as needed without requiring human interaction.
- *Broad network access*: The consumers can access the provisioned services over the network as and when required using various devices such as mobile phone, tablet, laptop, and workstation.
- *Resource Pooling*: The providers pool computing resources to serve users using multi-tenant model with physical and virtual resources assigned and released

dynamically according to consumers' demands. The users do not have the knowledge about the exact location of the provisioned resources.

- *Rapid Elasticity*: The resources are automatically provisioned or released commensurate with the demand. The computing capabilities available for provisioning often appear unlimited to Cloud consumers and can be appropriated in any quantity at any time. The users are not required to plan well before time of resource requirement. Further, the users are relieved from the risks associated with over provisioning or under provisioning of resources which otherwise leads to under-utilization or over-utilization of resources, respectively.

- *Measured Service*: Cloud computing has the capability to meter the usage of provisioned services. The provisioned services are monitored, controlled, and reported for transparency. The users are charged for what service (computing, storage, bandwidth) they use and the time they use the provisioned service.

The next section discusses the key foundation technologies of Cloud computing.

1.1.4 The Key Foundation Technologies

Cloud computing is the result of amalgamation of many computing paradigms including service-oriented architecture, grid computing, parallel computing, utility computing, autonomic computing, and virtualization. In many cases, it becomes difficult to distinguish between any of these technologies and Cloud computing. The relationships of Cloud computing with these technologies became more and more complex with the advancement in these technologies. The goal of this section is to explain how Cloud computing emerged from these technologies, and to clarify where these technologies stand within the context of Cloud computing. The key technologies which are considered as foundation of Cloud computing are:

1.1.4.1 Service Oriented Architecture

Service Oriented Architecture (SOA) is an architectural style which supports service orientation, a way of service based development [20]. A software based on SOA model

consists of loosely coupled, autonomous, stateless and fully documented services that can be accessed over the network. A service is a self contained unit of business activity with a distinct outcome which can be operated and updated independently. These services communicate and co-operate over the network to jointly provide the functionality of a large software application [21]. Web services standards are used for the implementation of services [22]. Cloud computing supports service orientation to enable global access, and ease of integrating different services and resources at run time. Similar to SOA, Cloud services can be accessed leveraging network-based software through standardized interfaces [23]. Cloud services are based on Representational State Transfer (REST)-style architecture [24], and support scalability and guaranteed quality of service. SOA is an umbrella that describes any kind of service [25]; however, a SOA service is not necessarily a Cloud computing service. Cloud computing uses rich Internet applications, Mashups, Asynchronous JavaScript and XML (AJAX), RSS to define, discover, and implement Cloud services.

1.1.4.2 Virtualization

Virtualization is a technology for dividing the server resources into multiple virtual execution environments. It provides a layer of abstraction between the hardware and the operating systems running on it [26]. The software or firmware that helps create this layer of abstraction on the server hardware is called a hypervisor or Virtual Machine Monitor (VMM). The advantages of virtualization are realized with the development of multi-core systems, clusters, grid computing and Cloud computing. It is considered as a better solution to optimize capital and operational expenditure by decreasing the number of physical servers by logically dividing each server and running multiple virtual servers on it. This is also known as server consolidation [27]. The virtualization vendors often use the term *"physical server"* to differentiate between a virtual server and a physical machine (server) [28]. A physical server implies the hardware that does the actual processing. According to Popek and Goldberg [29], *"A virtual server or Virtual Machine (VM) is an efficient, isolated duplication of a real machine"*. Thus, a VM is the software version of a physical server comprising an operating system and a set of applications. A VM is typically comprised of either a single file or a group

of files that can be read and executed by the virtualization layer. The virtual servers are also called the guests. Furthermore, virtualization isolates VM instances from one another and thus makes them unsusceptible to failures or hardware changes. An instance assumes that it is the only instance on the physical server. Virtualization decouples virtual server from the hardware underneath and thus makes it independent of a specific configuration of hardware to function. Virtualization enables the centralized control of resources while improving reliability and overall resource utilization. It allows creation of a snapshot of the VM and its storage devices at a particular time which can later be restored on some other server in case of emergency. Virtualization provides benefits including disaster recovery, and cost saving. It allows dynamic adjustment of the underutilized IT infrastructure resources as per the demands of the organization. IT infrastructure consists of a storage, network, server, operating system and application layer. The resources can be grouped together as a single logical unit, expanded or taken back in compliance with computation, storage or bandwidth requirement of the application. VM image can be cloned and get server up and running within minutes. This eliminates resource wastage by fully utilizing the physical server resources and by provisioning VMs with the exact amount of CPU, memory, and storage resources.

1.1.4.3 Grid Computing

Grid computing, a network of networks, is used to exploit a huge computational power. The constituent networks are usually located at different geographical locations. This large scale network of computers is used for complex computing operations. For instance, a popular Grid computing project "Folding@Home" utilized unused computing power of hundreds of thousands of physical machines to analyze protein folding and misfolding to understand the related diseases [30]. Workload in Grid computing, need to be known in advance, is distributed to the unused resources dynamically. The resources in a Grid are tightly coupled, thus a resource failure may have a cascade effect. Further, an individual user or Virtual Organization (VO) activity can impact the performance of other users using the same platform [31]; this will result in variable throughput and response time. Thus, it is difficult to guarantee Quality of Service

(QoS) in Grid computing [32]. The problematic issues of Grid computing are the use of real physical hardware and operating systems [33]. The initiatives to use virtualization with grid computing gave birth to "on-demand" Cloud computing, and addressed the reliability and QoS issues of grid computing [34]. Both virtualization and service oriented principles account for the paramount transition from the application oriented Grid computing to the service oriented Cloud computing.

1.1.4.4 Parallel and High-Performance Computing

Parallel computing is a computing paradigm where a complex problem is decomposed in many simpler computing tasks and distributed to multiple computing resources for concurrent execution [35]. The tasks and their related data is generally co-located on a physical server to ensure performance, and is usually referred to as principle of data locality [36]. Further, as parallel applications impose huge computing, so they are generally run on high-end servers which are very expensive.
The loosely coupled, transaction oriented, and latency insensitive parallel applications can be developed and deployed on Cloud using supported programming models. Many authors have also shown the suitability of Cloud platforms for data/compute intensive and scientific applications [37,38]. But it is not suitable for parallel applications that require complex communication patterns [37].

1.1.4.5 Utility Computing

The computational resources such as computation, storage, and services are leased to consumers as needed on per-per-usage basis rather than on flat rate. The consumers can manage the provisioned infrastructural resources. Utility computing aims to maximize the utilization of resources and minimize the associated costs. As the computational resources are leased, the users of this model need not bother about the upfront cost of the computational resources.

1.1.4.6 Autonomic Computing

Autonomic computing refers to the self-managing characteristics of distributed computing resources, and adaptability to unpredictable changes while hiding the intrinsic complexity from the operators and users. The adaptation process can be triggered by change in system state [39]. The self-management systems are developed to overcome the rapidly growing complexity of system management [40]. The system continuously checked and optimized automatically using high level policies. Autonomic computing enhances flexibility, increases resource availability and ensures optimal utilization. The architecture of autonomic systems is sometimes referred to as Monitor-Analyze-Plan-Execute (MAPE). Cloud computing is suitable for building autonomic system because it supports MAPE.

1.1.5 Cloud Ecosystem

Cloud ecosystem consists of agents involved in executing Cloud applications on the underlying IT infrastructure. The three main agents identified from the literature [39, 41–43] are Cloud users, Cloud aggregators and Cloud service providers which are discussed below:

- *Cloud Users*: Cloud users are the consumers of the Cloud services. They may be the end users using the Cloud hosted software services or may be the application providers requesting infrastructure services for hosting Cloud applications. With a pay for use pricing model, customers seeking a different type of Cloud services are billed by the service provider as per the usage terms. In addition, Cloud users agree to the service provider performance delivery specification, as stated in the SLA.

- *Cloud Aggregators*: Cloud aggregators serve as intermediary between the Cloud users and Cloud resource providers. They interact with the Cloud service provider on behalf of Cloud users. Cloud aggregators may request services from multiple service providers and manages the integration and delivery of diverse services to

Cloud users. They act as brokers to negotiate the service level agreements with the Cloud resource providers in support of Cloud users.

- *Cloud Service Providers*: Cloud service providers are the sellers of Cloud service. A service is packed in a server configuration as VM instance type. Each VM instance type differs from the others in terms of computation, memory and storage. Subsequently, a resource cost is also associated with the VM instance type. Users are charged per unit of time subject to the number and type of resource allocation. The amount of resource allocation varies during the life cycle of application execution.

After a brief introduction to the Cloud entities, the subsequent section presents the overview of Green Cloud computing.

1.2 Green Cloud Computing

A Cloud is a truly Green Cloud if it is powered by renewable energy, and provides customers with the interface for live monitoring of resources/services. Live monitoring helps to reduce the overall energy consumption of a data center comprising high performance computing infrastructure used for execution of scientific and engineering applications [44]. As per Wang *et al.* [44], a middle sized data center of a university consumes approximately 8000 kWh of energy. Further, the estimated energy usage of computing resources is about 0.5% of the world's total energy consumption [45], and is projected to quadruple by 2020, if current demand continues. Energy consumption of high performance facilities significantly contributes to operational expenses. According to Amazon's estimates at its data centers, 53% of the total budget accounts for expenses related to the cost and operation of the servers, out of which 42% is related to the energy required to run computational devices and associated cooling infrastructure [46]. Therefore, improving the energy efficiency in Cloud data center while respecting SLA can save significant energy budget for data center operators and would also make a substantial contribution to greater environmental sustainability. Thus, energy saving is one of the most critical challenges of the 21st century, calling

for efforts in a wide range of research fields, such as economics, engineering, ecological, and technical. In the Information Communication and Technology (ICT) area, energy aware initiatives are recently classified under the term Green computing. So, Green computing can be defined as an art and practice of designing, manufacturing, using and disposing computer resources in such a way that they have maximum efficiency and minimal impact on the environment. Green Cloud computing has witnessed many energy aware approaches and techniques, including VM consolidation, dynamic voltage frequency scaling, and efficient resource management, proposed by the research community and business industry [47–51].

1.3 Cloud Computing Adoption Challenges

The enterprises are adopting Cloud computing because of no upfront cost and economy of scale. But there are certain issues that need to be addressed to make the idea technically and commercially viable. Due to the increasing business agility, boosting flexibility and improving performance Cloud computing is gaining widespread adoption. Cloud computing is on its way to become a huge success. The several challenges related to growth and adoption of Cloud computing are:

- **Resource Allocation and Scheduling**: The resources are offered on pay per usage basis. Therefore, it it mandatory for Cloud service provider to optimize the allocation of resources with respect to heterogeneous and time variant environment in Cloud. The resource allocation system should respond with minimum time to sustain quality requirement [52].

- **Energy Efficiency**: The proliferation of Cloud computing has resulted in establishment of large sized data centers. The huge data centers consume vast amount of energy. So there is considerable concern about reduction in energy consumption and carbon footprints. The challenge may involve remodeling technologies employed at datacenters and energy aware resource management. The energy saving can be achieved through many strategies, one of them is consolidation of virtual machines to minimum number of servers and putting the idle

servers to sleep mode to conserve energy [53]. The goal is to cut down energy consumption of the data centers while meeting quality of service requirements of the end users. The key challenge is to reach a good trade-off between energy consumption and performance of the application.

- **Security and Privacy**: The security and privacy of data is of considerable concern for the organizations adopting for the Cloud. Cloud computing should ensure proper access and use of the data while maintaining its integrity [54]. The major security concern is that the consumer does not know the location of data. Cloud computing is lacking flexible, dynamic, and automated security management. The security policy should provide legitimate access and modification of the information [55]. It should also prevent illicit access and alteration of users' data.

- **Performance Unpredictability**: The Cloud consumers are unaware of the location of their data. This results in performance deviations when the processing module is not allocated resources in proximity to the data. Further, the input-output sharing problems aggravate the unpredictability in performance. So, there is a need to research the improved methods to efficiently virtualize the interrupts and input-output channels [56].

- **Service Lock-in**: Cloud computing provides Application Program Interfaces (APIs) for governance of services offered to the consumers/businesses. These APIs are proprietary and have no standardization across providers. Further, the Cloud providers offer services at cheap rate to retain the consumer till he shifts his core business requirements to the Cloud. This increase the risk of service lock-in because relying too much on a Cloud provider make it difficult to either move back to the original system or shift to some other Cloud provider due to proprietary software/applications that are platform dependent. With Cloud computing, switching of data/services swapping and integration is highly complex. Thus, there is a need to standardize the applications/software/APIs to mitigate these problems for better portability.

- **Service Reliability and Availability**: The Cloud consumers expect high reliability and availability of services from the Cloud providers. The complexity

of Cloud computing poses new availability and reliability challenges. The complexity adds more areas for potential failures. Thus, there is a need to recognize the area where the points of vulnerability lie. The new usage models offered by Cloud computing increases the chances of service failure. Service outages cause heavy loss to the customer. The loss is generally shared between consumers and service providers. There is a need to identify the contributing factors to these losses and risks. Further, well designed architecture and automation must be used to mitigate the risks.

- **Performance Bottlenecks**: Various types of bottlenecks are faced with Cloud computing. Bottlenecks cause not only performance loss but also results in loss of money for the customers. The bottleneck can occur in servers, network infrastructure, applications, and storage infrastructure. With Cloud computing services are provided to the consumers over the internet. The Cloud based applications are highly data intensive and demand high volume of data flow between the provider and the consumer. The large data flow hinders the adoption of Cloud computing. The performance bottlenecks can be alleviated through better resource management and efficient allocation techniques.

- **Licensing**: The traditional software licensing has not caught up with the flexibility and mobility of Cloud computing. One of the major Cloud adoption issues is the impact it has on licensing. It is a complex issue that is yet to be solved. Thus, there is a need to formulate a better licensing model for the Cloud. Generally, the software are licensed: per user, per device, or enterprise. The proprietary software licensing does not fit to dynamic Cloud environments. Due to this the Cloud providers rely on open source software. The dynamic capability of Cloud overwhelm the Cloud consumers with expensive software licensing cost.

This section summarized the key issues in the adoption of Cloud environment. The next section describes resource allocation in the Cloud environment which has been chosen as the area of research for this thesis.

1.4 Resource Allocation in Clouds: The Research Motivation

Resource allocation is a process of discovery, selection, deploying, and run-time management of resources in order to complete the hosted application as per negotiated quality of service constraints while meeting service providers' objective such as improved resource utilization, minimum energy cost and carbon footprints.

Figure 1.5 shows a resource allocation system that takes Cloud resources, and Qual-

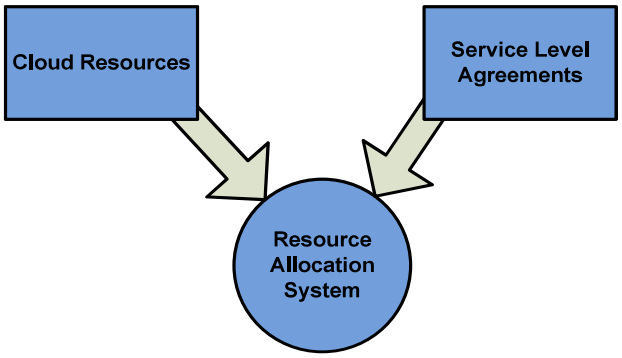


Figure 1.5: Resource allocation system

ity of Service (QoS) requirements in the form of Service Level Agreements (SLAs) as inputs to allocate resources to applications according to requirements.

The resources can be of two types: physical and virtual. The physical resources include CPU, memory, storage, and network components, whereas the logical resources comprise an operating system, energy, network throughput, protocols, and delays [57]. The resources are shared among multiple users, and are allocated to users' applications to provide services as per SLAs. These applications can be accessed by the users over the internet. As the end users of the applications are geographically dispersed, the application's workload fluctuates over time. Due to this, the data centers' infrastructure resources are generally over-provisioned to sustain the availability of resources during peak hours. But owing to the dynamic nature of load, average resource utilization remains approximately 15-20% [27, 58]. The major proportion of the total energy consumption of a data center is wasted due to under-utilization of resources because an idle resource consumes more than 50% of its maximum power consumption [59]. The

huge energy consumption of data centers is posing a big challenge to the researchers and the industry. According to Koomey [60], *"Total data center power consumption from servers, storage, communications, cooling, and power distribution equipment accounts for 1.7-2.2% of total electricity used in the U.S. in 2010"*. It had increased threefold between 2007 and 2012 [61]. Further, in 2013, data centers in the U.S. consumed an estimated 91 billion kWh of electricity, which was enough to power entire New York for two years [62]. And according to Kaplan *et al.* [58], energy consumption of an average sized data center is equivalent to energy consumption of 25,000 households. The huge energy consumption has increased Total Cost of Ownership (TCO), and has thus caused decrease in Return On Investment (ROI) for the Cloud service provider. Despite ROI reduction, energy consumption also causes carbon dioxide emissions, which is estimated to be 2% of global emissions [53]. Due to enormous energy consumption, today's data centers emit as much carbon dioxide as whole of Argentina. If left on their current path, data center carbon dioxide output will quadruple by the year 2020 [58]. The total carbon emissions in the world is estimated to be 1430 million metric tons by 2020, out of which data centers will account for 18% of the total emissions [63]. Figure 1.6 shows CO_2 emission growth by the year 2020. Koomey [60]

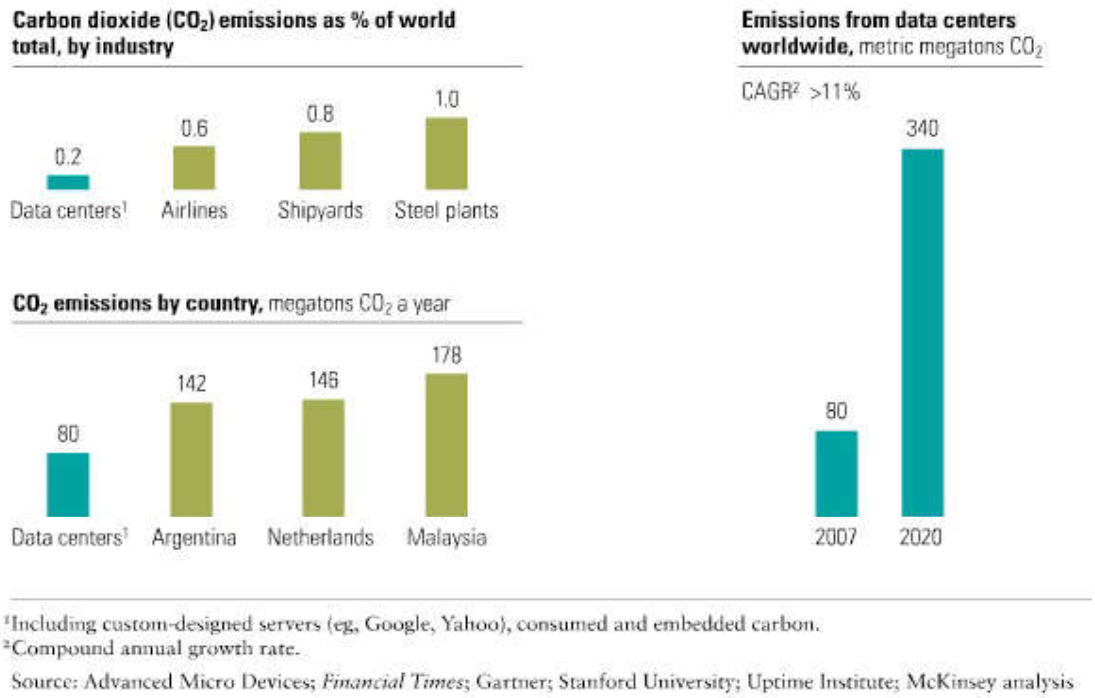


Figure 1.6: Datacenters carbon footprint worldwide [64]

estimates that energy consumption in data centers will continue to grow in the future. The huge energy consumption puts great pressure on the Cloud providers who need to reduce operational costs. While the Cloud energy consumption is growing quickly, industrial organizations and researchers are finding ways to reduce it.

On the other hand, under-provisioning of resources cause performance degradation [65] and consequently cause SLA violations. The consequences of performance degradation could be critical. One direct consequence may be losing users. For example, 100 ms delay resulted in 1% decrease in sales of Amazon, and Google observed 20% decrease in traffic with 0.5 second delay in search page generation [66].

Further, the Cloud service providers' objective is to maximize the utilization of resources through efficient resource allocation to earn maximum profit. This can be achieved by allocating applications to minimum number of physical servers which may cause SLA violations. However, the Cloud consumers wish to avail desired QoS at minimum cost. This demands dynamic reconfiguration of resources inline with fluctuating workload of the applications. Thus, allocation of appropriate resource capacity to an application is a challenging task from the energy and QoS perspective which is further aggravated by heterogeneity of resources, and heterogeneous and fluctuating workload.

1.5 Thesis Organization

The thesis is divided into seven chapters as discussed below:

Chapter 1: Introduction

The chapter starts with overview of Cloud computing, and then describes how it evolves from the various key computing technologies. Further, it presents the essential characteristics, various types of services and deployment models supported by Cloud computing paradigm. The chapter also introduces some of the key foundation technologies from which Cloud computing has evolved. Further, it elucidates Cloud adoption challenges, motivation to energy-efficient resource allocation, and organization of thesis into different chapters. The chapter concludes with thesis contributions.

Chapter 2: Literature Survey and Related Work

This chapter presents literature survey on energy efficient resource allocation in clouds. The categorization of the resource allocation approaches on the basis of technique used, adaptation policy, and QoS has been performed. The comparison of different approaches on the various factors has also been done. The state-of-the-art techniques and related approaches in the context of resource allocation has been presented. More specifically, heuristics and metaheuristic techniques for resource allocation have been analyzed. The chapter concludes with *Problem Formulation* on the basis of findings of the available literature on energy and QoS aware resource allocation.

Chapter 3: SERVmegh: A Green Cloud Framework

This chapter presents the proposed framework for Green Cloud for efficient and robust management of resources. The framework is layered in nature. In addition to a brief overview of all the six layers and their components, the chapter also presents comparison of the *SERVmegh* framework with various open-source as well as commercial Cloud frameworks. The mathematical expression for overhead of layered architecture is also derived. The resource wastage reduction methodology is devised for energy efficient resource allocation. Performance evaluation of the proposed approach has been carried out on simulated as well as OpenNebula based private Cloud environment.
The chapter is derived from [67]:

- **Ashok Kumar**, Anju Sharma, Rajesh Kumar "SERVmegh: framework for Green Cloud," Concurrency and Computation: Practice and Experience, ISSN 1532-0634, 2016. DOI: 10.1002/cpe.3903

Chapter 4: Energy Aware Resource Allocation for Clouds

This describes energy aware resource allocation approach for Clouds. The chapter starts with architectural overview of the proposed approach, followed by derivation of mathematical expressions for total energy consumption and execution cost. The chapter also discusses Dynamic Voltage Frequency Scaling (DVFS) for reducing energy consumption. The chapter further presents Ant Colony Optimization (ACO) metaheuristic for allocation of resources to tasks. ACO is employed at two levels. At first level, ACO is applied to allocate tasks to virtual machines. The virtual resources are allocated according to length of the jobs and their associated QoS requirements. At second level, ACO decides the placement of virtual machines over the physical

servers. The geographical distance of physical machines and available resource capacity is taken in to consideration for virtual machine placement. The chapter concludes with performance evaluation of the proposed work.
This chapter is derived from [68]

- **Ashok Kumar**, Rajesh Kumar, Anju Sharma "Energy Aware Resource Allocation for Clouds Using Two Level Ant Colony Optimization," Computing & Informatics, ISSN 1335-9150, 2016 [in press]

Chapter 5: EQUAL: Energy and Quality of Service Aware Resource Allocation

This chapter presents energy and QoS aware resource allocation approach that can be operated in three modes: power aware mode, performance aware mode, and balanced mode. When operated in power aware mode, the proposed approach strives to use minimum number of physical resources in order to reduce energy consumption. In performance mode, the applications are allocated to servers that have maximum available capacity at disposal. Whereas, in balanced mode, power and performance are given equal weightage while allocating resources to the applications. The mode of operation can be changed by varying the values of the control variables. The best resource for deployment of an application is discovered with Antlion optimization by mimicking random movements of Ants and Antlions using levy flight based random walk. The position of an Antlion in the search space refers to a solution which is progressively improved through random walk of Ants around it. The proposed approach is implemented in CloudSim simulator for performance evaluation and validation.
This chapter is derived from [69, 70].

- **Ashok Kumar**, Rajesh Kumar, Anju Sharma, *"EQUAL: Energy and QoS Aware Resource Allocation Approach for Clouds"*, Computing and Informatics, ISSN 1335-9150, 2016. [in press]
- Rajesh Kumar, **Ashok Kumar**, Anju Sharma, *"A Bio-inspired Approach for Power and Performance Aware Resource Allocation in Clouds"*, in: 4th International Conference on Advancements in Engineering and Technology (ICAET-2016), ISSN 1532-0634, 2016.

Chapter 6: A Self Optimizing System for Energy and Quality of Service Aware Resource Allocation

This chapter presents a self optimizing system for energy and QoS aware resource allocation that optimizes the allocation of resources in accordance with the workload. The self optimization resource allocation process is performed in four phases: *Monitor*, *Analyze*, *Plan* and *Execute*. The *Monitor* phase collects the utilization metrics of each virtual machine from the probes installed in physical machines and stores them in centralized performance data base for later processing. *Analyze* phase uses historical data for predicting the resource needs of the applications in the near future. The *Plan* phase decides about the mode of operation for the next time slot from the predicted resource needs. The decision about migration of virtual machines from heavy loaded physical machines is also taken in this phase. The decisions taken in *Plan* phase are applied in *Execute* phase. The necessary changes are made in the control variables to shift the resource allocation system to different mode of operation that has been inferred in *Plan* phase. The VMs are migrated to offload heavily loaded machines or consolidate lightly loaded machines to fewer physical machines to save energy by switching idle machines to low power modes.

Chapter 7: Conclusion and Future Scope

The thesis concludes with a brief overview of the proposed Green Cloud framework, energy and QoS aware resource allocation, and self optimization resource allocation system. This chapter also shed light on open issues and future perspectives in resource allocation.

1.6 Thesis Contributions

The contributions of this thesis can be broadly classified into five major categories: analysis and classification of related research work, a framework for green cloud, energy aware resource allocation, energy and QoS aware resource allocation and self-optimizing resource allocation system. The key contributions of this thesis grouped under these five categories are:

1. Analysis and Classification of Related Research Work

- A taxonomy and comparison of the existing approaches related to energy efficient and/or QoS aware resource allocation has been presented.
- A brief overview of energy and/or QoS aware approaches to resource allocation in clouds has been provided.

2. Framework for Green Cloud

   - A green cloud framework for efficient and robust management of resources has been proposed, and its different components have been elaborated.
   - Comparative analysis of the proposed framework with existing open-source and commercial cloud frameworks has been carried out.
   - Mathematical expression for the overhead of layered green cloud framework has been derived.
   - The prototype implementation of *resource management*, *green power management*, *workload analyzer and manager*, and *on/off control* components of proposed cloud framework has been done.
   - A novel energy efficient resource allocation approach based on resource wastage reduction methodology is developed.
   - A private cloud based on OpenNebula open-source cloud environment has been set-up with 40 physical nodes to support the deployment and execution of heterogeneous virtual machines. The proposed resource allocation approach is validated on this private cloud.

3. Energy Aware Resource Allocation

   - A novel two stage energy aware resource allocation approach is devised.
   - Ant colony optimization is applied for optimal allocation of resources. Geographical locations of data centers, length of jobs, and QoS attributes (cost and energy consumption) of jobs are considered while allocating resources.
   - Energy consumption is reduced by switching idle machines to sleep mode and dynamically scaling CPU performance through adjustments in voltage and frequency of the processor.

- The proposed approach is compared with state-of-the-art energy aware resource allocation approaches in clouds.

4. Energy and QoS Aware Resource Allocation

   - Energy and QoS aware resource allocation approach is presented that can be tuned to: power aware, performance aware, or balanced mode of operation.
   - A meta-heuristic (Antlion optimization) is employed for optimal allocation of resources to tasks.
   - The proposed approach is validated in simulated cloud environment.
   - 15.04% reduction in energy consumption is achieved.

5. Self-optimizing Resource allocation System

   - A Self-optimizing resource allocation system has been proposed that automatically switches to energy aware mode or performance aware mode depending on the workload of the applications.
   - The utilization metrics history is used to predict mode of operation.
   - Performance analysis of the proposed work is carried out.

Chapter 2

Literature Survey and Related Work

Cloud computing is a computing paradigm which offers various services on pay-per-usage basis over the internet. Its alluring features such as scalability, high availability, accessibility, cost savings, and on demand self service, have motivated the service providers to adopt and offer services to users via data centers. The growing popularity of Cloud computing encouraged providers to increase underlying computing infrastructure of their data centers to accommodate service demand of the users. These large scale data centers have caused a drastic increase in energy consumption and carbon emissions. The ever growing energy consumption of these data centers has become an important challenge for researchers and Cloud service providers. As Cloud computing is outgrowth and amalgamation of previous distributed systems, It requires novelty in resource allocation methods, techniques and approaches. In order to identify challenges and facilitate further advancements, there is a need to classify and synthesize the research in this area.

In this chapter, an extensive survey on energy-efficient resource allocation techniques existing in the literature till date has been conducted to understand the issues pertaining to energy efficient resource allocation under Quality of Service (QoS) constraints. This chapter begins with basic concepts covering Green Cloud computing, power and energy, server power consumption, and modeling power consumption. It then presents power management techniques, QoS in Clouds, and resource allocation along with its related terminology. The chapter then presents extensive survey on resources allocation techniques along with their comparison. Finally, the chapter concludes with limitations of the existing techniques and outlines the objectives of the thesis.

2.1 Green Cloud Computing

Energy consumption of the data centers is growing year by year due to proliferation of Cloud computing and subsequent increase in computing infrastructure. Data centers are gradually becoming the fastest growing power consumers. The annual estimated energy consumption of U.S. data centers is 91 billion kWh which is equivalent to electricity generation capacity of about 34 coal-fired power plants, each having production capacity of 500 mega-watts. This annual energy consumption is estimated to touch 140 billion kilowatt-hours mark by 2020, equivalent to the output of 50 such power plants [62]. The huge energy consumption demands additional cooling system and power supply hardware. A major proportion of the huge energy required to power these data centers is wasted due to the inefficient allocation of computing resources. The energy wastage not only causes monetary loss but also poses potential environmental threat, as it releases CO_2 in the environment. Therefore, improving the energy efficiency in Cloud data center while respecting Service Level Agreement (SLA) can save significant energy budget for data center operators and thus make a substantial contribution to greater environmental sustainability. Therefore, energy saving is one of the most critical challenges of the 21st century, calling for efforts in a wide range of research fields, such as economics, engineering, ecological, and technical. In the Information Communication and Technology (ICT) area, energy aware initiatives are recently classified under the term Green computing. A Cloud is a truly Green Cloud, if it is powered by renewable energy. Green computing aims to attain economic viability and improves the way computing devices are used. So, Green computing can be defined as an art and practice of designing, manufacturing, using and disposing computer resources in such a way that results in maximum efficiency and minimal impact on the environment. To promote Green computing concepts at all possible levels, the following four complementary approaches are employed:

- *Green Use*: Minimizing the electricity consumption of computers and their peripheral devices and using them in an eco-friendly manner.
- *Green Disposal*: Re-purposing an existing computer or appropriately disposing off or recycling unwanted electronic equipment

- *Green Design*: Designing energy-efficient computers, servers, printers, projectors and other digital devices.
- *Green Manufacturing*: Minimizing waste during the manufacturing of computers and other subsystems to reduce the environmental impact of these activities

The main focus of this thesis is to make Cloud "Green" through *Green use* of resources. For this, energy efficient resource allocation techniques have been developed is to curb energy consumption and hence the carbon emissions to the environment while respecting QoS requirements of the end user. Green Cloud computing has witnessed many energy aware approaches and techniques. The existing energy-aware techniques cannot be grasped without the clear understanding of the basics of power and energy. Therefore, a brief overview of these is given in the next subsection.

2.1.1 Power and Energy

The flow of electric charge through an electric circuit per second is referred as current, and it is measured in amperes. The power applied to Physical Machine (PM)/server and its energy consumption can be expressed in terms of amount of work that it performs. The rate at which a server performs a task is regarded as power. Whereas, the amount of work done per unit time is called energy. The units of measurements of power and energy are watt (W) and watt-hour (Wh), respectively. For example, if 1A current is passed through a potential difference of 1V then work is performed at the rate of 1W. Further, one kilowatt-hour (kWh) energy is consumed if 1kW (1000W) of power is applied for one hour. Power and energy can be determined from equations (2.1) and (2.2).

$$P = W/T, \tag{2.1}$$

$$E = PT, \tag{2.2}$$

where W is the work done, and E is the energy consumption when power P is applied for time period T.

2.1.2 Server Power Consumption

Total power consumption of a PM/server is given by Beloglazov *et al.* [71] as shown in equation (2.3)

$$\begin{aligned} P_{total} &= P_{dynamic} + P_{static} \\ &= \underbrace{ACV^2f}_{\text{DPC}} + \underbrace{V * I_{leak}}_{\text{SPC}} \ , \end{aligned} \tag{2.3}$$

where A is switching activity, C is capacitance, V is voltage, I_{leak} is the leakage current, and f is the clock frequency applied to the cores of PM. P_{total}, P_{static}, and $P_{dynamic}$ are total power consumption, Static Power Consumption (SPC), and Dynamic Power Consumption (DPC) of a PM, respectively. SPC is due to leakage current that is present in any active circuit independent of clock frequency and usage scenario. It can be reduced by making changes in the hardware circuitry that is beyond the scope of this thesis. Whereas, DPC is due to circuit activity and it depends on usage scenario or resource utilization.

2.1.3 Modeling Power Consumption

To develop a novel energy efficient technique and understand its impact, it is necessary to model dynamic power consumption of a server as a function of its resource usage. Such a model should be able to find the power consumption of the server from its run-time characteristics, which can be collected using monitoring capabilities that are built into modern servers. Based on run time performance metrics data, it is possible to derive a power consumption model for a particular system. However, this approach requires collecting data for every target system. Fan *et al.* [72] gave a power consumption model based on CPU utilization. According to this model, power consumption of a server has a linear relationship with its utilization. That is power consumption of a server increases linearly with the increase in it utilization. The server consumes some power even when it is idle. The power consumption of the server increases from the power consumption in idle state to the maximum power consumption when the CPU is fully utilized. The relation between power consumption

and CPU utilization is given by [72]:

$$P(u) = P_i + (P_m - P_i)u, \tag{2.4}$$

where P is the projected power consumption, P_i is the power consumption of the server when it is idle, P_m is the maximum power consumption of the server at full utilization, and u the CPU utilization at any moment. Using the linear model power consumption can be predicted with error below 5%. More accurate empirical nonlinear power model is given by Fan *et al.* [72], that can predict the power consumption with error less than 1% which is given as:

$$P(u) = P_i + (P_m - Pi)(2u - u^r), \tag{2.5}$$

where, the parameter r is obtained for each class of machine by experiment that minimizes the square error. Using these models, power consumption of a server for different levels of utilization can be easily obtained from the data comprising power consumption at different CPU utilization levels which is usually published by the server manufacturers [73].

The next section discusses the power management techniques that are applied at different levels to reduce energy consumption.

2.2 Power Management Techniques

The energy consumption of data centers is largely due to servers. Power consumption by different components of a server is depicted in Figure 2.1. It is clear that CPU and memory of a server are the major power consumers. According to Meisner *et al.* [74], average power consumption of CPU and memory of a server is about 56% of its total power consumption. Therefore, a technique that reduces energy consumption of CPU and memory would surely improve total energy consumption of the server, and hence the energy efficiency of data center. A lot of research has been done in the area of power management in computing systems. Power management techniques can be applied at hardware level, operating system level, virtualization level and data center level as depicted in Figure 2.2.

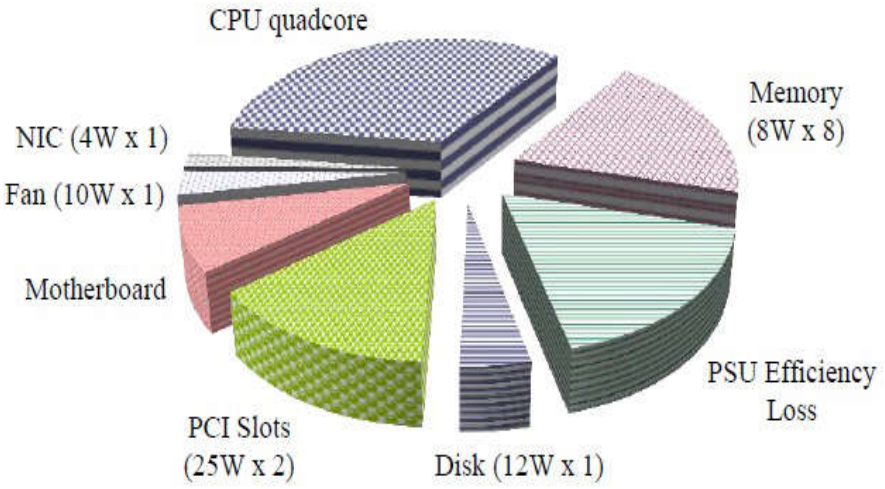


Figure 2.1: Power consumption by server components [75]

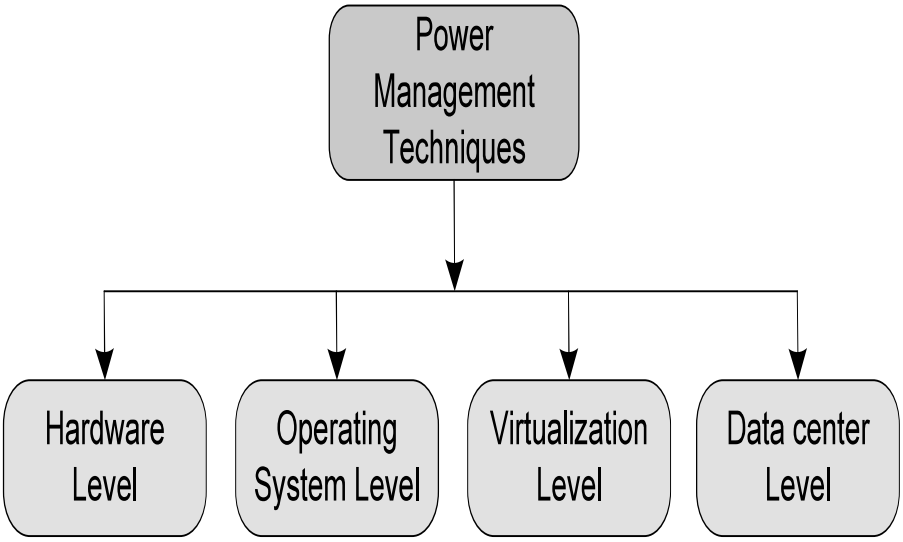


Figure 2.2: Taxonomy of power management techniques

2.2.1 Hardware Level

Power consumption of a computer system comprises static and dynamic power consumption as discussed in Section 2.1.2. Therefore, power managements methods can be classified as:

- *Static Power Management*: Static Power Management (SPM) techniques are applied at design time and built in the hardware at the time of manufacturing. SPM techniquess can be applied at circuit, system, or architectural level [76,77].

- *Dynamic Power Management*: Dynamic Power Management (DPM) techniques are applied at hardware level to reduce DPC of the servers. This technique can be further classified into two types:
  - *Dynamic Component Deactivation*: In dynamic component deactivation technique, a component or the system is partially or completely disabled during the period of inactivity. It is generally applied for those components of the computer which do not support performance scaling, and is viable when the power and performance overhead of switching between states is negligible.
  - *Dynamic Performance Scaling*: In dynamic performance scaling, performance of a component is adjusted inline with the real-time demands. It can be applied to those components of a PM which support adjustment of performance proportional to supplied power. For instance, modern-day CPUs support Dynamic Voltage and Frequency Scaling (DVFS), a gradual increase or decrease of the clock frequency in combination with supply voltage, to adjust the level of performance. This approach is beneficial when a component is not being used at full capacity. DVFS is also employed to control the amount of heat produced by the components [78]. AMD Turbo Core [79], and Intel Turbo Boost [80] are the technologies which support dynamic adjustment in operating frequency and voltage of the CPU to conserve energy.

2.2.2 Operating System Level

Operating System (OS) can be customized to support power management in real time, so that accurate accounting of energy consumption by the processes can be made over all the resource. For the purpose of power management, OS is structured in such a way that the users can directly call system functions, protocol stacks, and device drivers in their application programs.

Neugebauer and McAuley [81] developed an operating system for battery powered devices named "Nemesis OS" which provided fine grained control over the various

components of the underlying hardware such as CPU, memory, disk and network to support power management in line with the workload. The OS was vertically structured to provide per process resource usage and energy accounting. Another OS based power manager called "On-demand governor" was presented by Pallipadi and Starikovskiy [82]. On-demand governor was implemented in the Linux OS and could manage the power consumption in real-time by varying the clock frequency and supply voltage inline with the performance requirements. Meisner *et al.* [74] developed an energy-aware OS that exploited fast transitions between active and low-power states to reduce energy consumption by the server. The proposed approach utilized only two power states for each component of the server i.e. operating state and sleep mode. Rajkumar *et al.* [83] customized Linux kernel with the goal to minimize energy consumption, and performance isolation between processes. The modified Linux kernel was named as "Linux/Resource Kernel" and it used DVFS based algorithms for energy minimization.

2.2.3 Virtualization Level

Virtualization is a technology that has capability to improve utilization and hence the energy efficiency of server resources. It creates a layer of abstraction between the hardware and the operating system [26]. The software or firmware that helps create this layer of abstraction on the server hardware is called a hypervisor or Virtual Machine Monitor (VMM). This layer helps divide the server resources into multiple virtual execution environments. Each virtualized environment can accommodate a virtual server or Virtual Machine (VM) that is a software version of a physical server comprising an operating system and a set of applications. Virtualization is viewed as a solution to optimize capital and operational expenditure by decreasing the number of physical servers through logically dividing each server and running multiple VMs on it. The physical resources of the servers are under the direct control of VMM. Therefore VMM can help manage the power consumption by monitoring the system resources in real-time and applying appropriate power management technique. For instance, Xen supports frequency scaling through *cpufreq* driver [84]. It analyzes the periodically monitored CPU utilization and then make changes in the power state of the hardware.

Xen also have inbuilt support for sleep states, called C-states [84], therefore, Xen can switch underlying server to sleep mode when it is idle and bring it back to active state as the new task arrives. VMM can also be implemented as module of Linux kernel. In this case, the OS treats VMs running over the VMM as processes, and applies power management instruction issued by encapsulated VMs on the underlying hardware. For example, Kernel-based Virtual Machine (KVM) is implemented as a module of the Linux kernel [85]. KVM also supports the hibernate and sleep power states for dynamic power management [86].

The next section discusses research efforts in the area of power management at the data center level.

2.2.4 Data Center Level

With Cloud computing, the services are delivered to the users on pay-per-usage basis. Maximizing the profit is the top priority of the service providers. In this regard, operational expenses including energy consumption plays a crucial role. The provider's profit share increases with reduction in energy consumption. Recursively, energy consumption can be reduced by increasing the utilization of resources. The allure of profit making motivate the service providers to make every effort to reduce energy consumption. Therefore, a service provider typically implement better power management techniques.

The energy consumption at data center level can be reduced by consolidating the applications to fewer physical servers and deactivating the idle servers. This helps to improve utilization of resources and energy efficiency of the data center. Consolidation of applications is a challenging task as it may lead to performance degradation of the applications if applied aggressively. Performance degradation may lead to increased response times, timeouts, or failures. Therefore, to keep the balance between energy consumption and performance, power saving techniques are applied under constrains of QoS requirements of the applications.

Pinheiro *et al.* [87] proposed "load concentration or unbalancing" based technique for managing power consumption while providing required QoS in a non-virtualized cluster of PMs. The main idea is to concentrate workload on lesser number of servers and

switching off the idle servers to conserve energy. The approach causes performance gradation of application due to consolidation.

Chase *et al.* [88] developed an OS, Muse, for managing resources of internet hosting centers with the aim to minimize energy consumption while respecting QoS requirements of the applications. The applications have QoS requirements in terms of throughput and latency constraints, and shares the hosting servers. The proposed OS strives to reduce operational expenses, CO_2 emissions, thermal susceptibility of the system because of failure of cooling system, and over-provisioning in capacity planning. Muse supports flexible SLAs because of trade-off between the service quality and price. The authors' work has proved as a foundation stone for research in the area of power-efficient resource management, but it suffers from some weaknesses. It manages only CPUs, the other resources of the servers are not taken care of. The authors have also not considered the thermal factor and latency of switching physical nodes on/off.

Chen *et al.* [89] developed an approach for provisioning of multiple applications in hosting centers while minimizing energy consumption and respecting QoS requirements. The authors considered response and cost as QoS parameters. The approach works in two phases that are executed periodically. In the first phase, number of servers are allocated to the application as per its current workload level. The second phase applies DVFS technique to set the performance of servers for executing application's current workload with minimum energy consumption. The energy consumption is reduced by switching off idle servers after each allocation phase, and by applying DVFS to each server.

Srikantaiah *et al.* [90] proposed dynamic consolidation approach to minimize energy consumption by data centers. The authors studied the impact of workload consolidation on the energy consumption of applications. It is observed that low utilization leads to higher cost in terms of the energy-performance metric. Whereas, increased cache miss rate, context switches, and scheduling conflicts are observed when the resource utilization is high. Further, high utilization causes high energy consumption, performance degradation, and consequently longer execution time.

Garg *et al.* [91] studied energy and CO_2 efficient scheduling of high-performance computing (HPC) applications in order to provide user high end computing resources

offered by the Cloud on pay per usage basis. The proposed heuristics perform energy-efficient and carbon aware meta-scheduling of applications across heterogeneous data centers considering energy cost, carbon emission rate, workload, and CPU power efficiency. They proposed scheduling policies to minimize CO_2 emissions and maximize the profit of resource providers. The authors also employed DVFS to reduce energy consumption of data centers.

The service providers earn profit by delivering services to the end users, therefore providing better QoS to the end users is one of the most important obligations of the service providers. So, power management techniques for energy efficiency are required to be applied under QoS constraints. The next section discusses QoS requirements that become critical for service providers to oblige.

2.3 Quality of Service in Clouds

Quality of service refers to the service performance observed by the Cloud service consumers. It is the ability to provide different priority to different applications. It also refers to a guaranteed level of service quality. QoS is measured quantitatively by several related aspects of the Cloud. QoS requirements in Cloud environments are stated in the SLA and differ largely from the traditional computing paradigms. The QoS parameters are usually interrelated because of trade off between them. Type and level of service quality required varies with the kind of task submitted to Cloud for execution. For example, tasks related to online transaction processing require security, high availability, accessibility and usability. The various metrics that collectively contribute to the quality measurement of Cloud services are described below:

- *Timing Constraint*: The timing constraint specifies the acceptable limit of time for the execution of users' applications. An application's timing constraint can be specified in terms of deadline, advance reservation or best-effort. The execution of the deadline constraint applications must finish by the maximum time specified by the client. With advance reservation application, the launch time

of the application follows the predefined timings. The application with best-effort timing constrained are not guaranteed timely completion, but generally provisioned as per their peak demand.

- *Performance*: Performance is an abstract QoS property which includes sub-properties like throughput, elapsed time, and response time. *Throughput* is the number of tasks or transactions processed per unit time. It is measured in transactions per seconds for web applications. The amount of time taken to complete the end user request is termed as *elapsed time*. *Response time* represents the time period between submission of request and receipt of the response.

- *Reliability*: Reliability refers to the ability of a software/hardware component to perform desired function for a certain period of time under some restrictions. Cloud data centers comprise tens of thousands of servers which coordinate to execute end users' tasks. The failure of a server in such a large scale data center is the norm rather than an exception [92]. Therefore, providing high reliability is a challenging job in Cloud computing. Reliability is generally measured in terms of Mean Time to Failures (MTTF) and Mean Time to Repair (MTTR) as given by Singh and Chana [93] from equation (2.6).

$$Reliability = MTTF + MTTR, \tag{2.6}$$

  where MTTF is the expected mean time until the first failure of the system, and MTTR is the mean time needed to repair a failed system.

- *Availability*: Availability is the proportion of time a system or a component is functional and committable. As per Beloglazov *et al.* [94] availability can be evaluated as:

$$Availability = \frac{MTBF}{MTBF + MTTR}. \tag{2.7}$$

- *Cost*: Cloud supports pay as you go model where cost of availing a service is determined from resource acquisition and utilization levels. Primarily CPU, storage and network bandwidth are considered for cost evaluation which have fixed acquisition cost and variable usage cost. The usage cost is measured in

terms of the resource utilization levels which is equivalent to the proportion of time a resource is busy in servicing user requests excluding idle time [95].

- *Energy Efficiency*: Energy efficiency is emerging as a key metric in resource allocation for Cloud environments. Cloud service provider can increase his share of profit by opting energy efficient resource allocation techniques which generally employ dynamic component deactivation, server consolidation, or DVFS mechanisms. According to Beloglazov *et al.* [94], energy consumption by a server over a period of time $(t_1 - t_2)$ can be evaluated from the integral of power consumption using equation (2.8).

$$E = \int_{t_1}^{t_2} P(t)dt, \tag{2.8}$$

  where E is the energy consumption over the period $(t_1 - t_2)$, $P(t)$ is the power consumption at any instant of time.

- *Security*: The security and privacy of data is of considerable concern for the organizations adopting for the Cloud. Cloud computing should ensure proper access and use of the data while maintaining its integrity [54, 96]. The major security concern is that the consumer does not know the location of data. Cloud computing is lacking flexible, dynamic, and automated security management. The security policy should provide legitimate access and modification of the information [55]. It should also prevent illicit access and alteration of users' data. The past experience with a service provider can be used as a parameter to compute the trust value of the service provider.

Energy efficient resource allocation in Green Clouds is the research motive in this thesis. The next section discusses about resource allocation in Green Clouds.

2.4 Resource Allocation in Green Clouds

Resource allocation system is an indispensable part of Cloud environment. It should allocate the resources to applications in such a manner that minimizes energy consumption while respecting QoS requirements of the end users. The applications are

encapsulated in VMs which may be differently dimensioned in terms of number of CPU cores, memory, and storage requirements. Each application is also associated with QoS demands. The applications are offered as services to the end users. The resource allocation system should allocate physical resources to the services/VMs to minimize energy consumption under QoS constraints.

The approaches existing in the literature on resource allocation can be broadly classified into two types: offline and online/dynamic. In offline technique, the resource allocation system have complete knowledge of the applications' requirements as well as the available resources/PMs. The resources are allocated to all the applications ahead of time. The applications are then scheduled for execution on the predetermined resources without considering newly arriving service requests and fluctuating demands of existing applications. However, dynamic allocation technique considers dynamism in the Cloud environment. It considers the fluctuating resource demands of the applications and accordingly adjusts the resources allocated to the applications.

Resource allocation is investigated as an NP hard problem [109][110], so the solution of the same cannot be determined in polynomial time [111]. The complexity of the problem further worsens due to the presence of multitude of Cloud applications, users, QoS criteria and resource usage scenario. Many researchers are working to circumvent the problem of Cloud resource allocation. However, the focus of their research is mainly targeted towards either a single application type (HPC, batch application or transactional workloads) or addressing a specific resource type (bandwidth, CPU or memory) with a single QoS metric (deadline for HPC and response time for transactional applications) mentioned in SLA.

Resource allocation taxonomy given in Figure 2.3 is the outcome of the survey on resource allocation.

An extensive survey on the existing resource allocation techniques is conducted on the basis of the parameters discussed in the following subsections.

2.4.1 Adaptation Method

Cloud is dynamic in nature. The workload of the applications/services keeps changing with time. Therefore, resources allocated to an application need to be adjusted

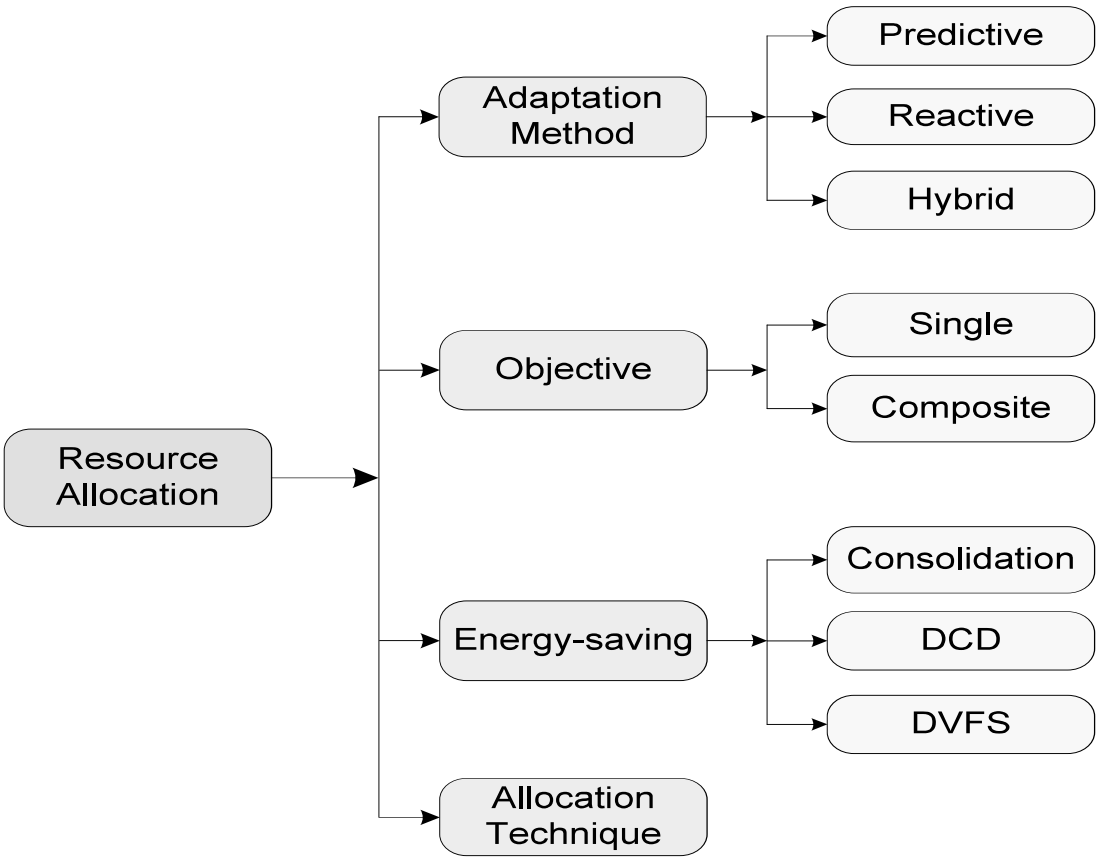


Figure 2.3: Taxonomy of resource allocation

with time. The process of adjusting/reconfiguring of compute, memory, network or storage resources allocated to Cloud applications inline with the change in workload is termed as adaptation. The adaptation process causes change in data center revenue, data center power consumption, number of active servers or end-user experience. The adaptation is driven by some objective that decides the resources to reconfigure. For instance, a VM can be adapted by adjusting the resources allocated to the VM in order to accommodate the change in the workload by adjusting CPU, memory, bandwidth and/or storage resources allocated to it. Similarly, a node can be adapted to lower power consumption by employing DVFS, and a cluster can be adapted through dynamic component deactivation. Based on the existing literature, three types of adaptation methods have been identified:

- *Predictive*: Predictive adaptation is a widely applied method. It uses a model to predict the system's behavior over a finite future horizon and chooses the

control action that optimizes a cost function subject to some constraints. Predictive adaptation method can be applied when sufficient data pertaining to resource performance and workload is available, and historical data follows some distribution. Further, predictive adaptation is expensive in terms of storage and processing cost of historical data.

- *Reactive*: Reactive adaptation is based on a hard-coded and pre-decided corrective measure which is triggered when a specific event occurs, such as utilization of the CPU reaches certain threshold, or energy consumption of a CPU goes beyond threshold. The efficiency of the reactive allocation is judged from its ability to detect fluctuations. Resource allocated to the applications need to be adjusted inline with the change in the workload, in order to minimize deviation from the QoS performance goals [97]. Reactive method does not require extensive knowledge about application's real time behavior. It suffers from issues such as high provisioning cost [98].

- *Hybrid*: In hybrid adaptation method, both predictive and reactive adaptations are combined for: (i) meeting SLA, (ii) energy conservation, and (iii) reduction in provisioning cost. Xie and Wilamowski [99] showed that coordinated management between predictive and reactive approaches results in a significant improvement in energy efficiency without sacrificing performance.

2.4.2 Objective

Resource allocation technique has some goals to achieve. For instance, reducing the overall energy consumption of the infrastructure can be objective of a resource allocation technique. The objective can be single or composite based on the number of factors considered. The aim of minimizing energy consumption is a single objective. Whereas, minimizing both energy consumption and response time at the same time is composite objective. The objective function is generally a mathematical expression which is desired to be minimized or maximized.

2.4.3 Energy-saving

Data centers are growing as the major energy consumers. This is because of allocation of resources as per peak load of the applications. The following energy-saving methods have been identified from the existing literature.

- *Consolidation*: Consolidation refers to the use of a physical server to accommodate one or more server applications or user instances. Consolidation works on the principle of server virtualization, where one or more virtual servers reside on a physical server. With the use of consolidation, it is possible to share a server's resources among multiple applications and services at the same time. The primary objective behind consolidation is to increase the utilization of a server resources, and reduce the associated capital and operational expenses. Traditionally, only 15-30 percent of a physical server's overall capacity is used. Due to multi-tenant architecture of Clouds, the virtual servers can share processor, storage, memory and network resources. The immediate benefits of consolidation are: less hardware requirements, reduction in data-center footprint, and reduction in power consumption. Reducing power consumption is an increasingly important driver for consolidation since energy companies has started to provide significant incentives for cutting energy consumption.

- *Dynamic Component Deactivation*: In dynamic component deactivation approach, a component or the system is partially or completely disabled during the period of inactivity. It is generally applied for those components of the computer which do not support performance scaling, and is viable when the power and performance overhead of switching between states is negligible.

- *Dynamic Voltage Frequency Scaling*: Dynamic voltage frequency scaling is a technique to save energy consumption by intentionally scaling down the performance of CPU by setting the voltage and frequency to some lower supported value in combination during the periods of low activity.

2.4.4 Allocation Technique

Energy consumption in hosting centers and server farms is rapidly increasing [58, 60–62]. It can be curbed by applying energy efficient resource allocation techniques [52]. A variety of resource allocation techniques have already been devised, but there is scope for further improvement. Taxonomy of resource allocation techniques is depicted in Figure 2.4. To identify the gaps in the existing resource allocation techniques, extensive survey of the techniques has been carried out in the following section.

2.5 Resource Allocation Techniques

The techniques existing in the literature have been classified in the following categories on the basis of approach used in resource allocation.

2.5.1 Bio-Inspired Resource Allocation

Feller *et al.* [100] presented multi-dimensional ant colony optimization based job consolidation algorithm. The algorithm uses resource utilization history to predict future resource demands and dynamically overbooks the resources. The proposed algorithm is validated on homogeneous PMs. Huang *et al.* [101] presented genetic algorithm based adaptive sub-optimal resource management scheme to estimate number of VMs required to provide desired level of service. Kansal and Chana [102] suggested a model based on artificial bee colony to improve utilization of resources. The model supports energy efficient allocations of tasks to resources while minimizing execution time of applications. Chimakurthi and Madhu Kumar [103] offered ant colony based adaptive resource allocation framework for hosting applications with throughput and response time as QoS requirements. Further, it supports reduction in power consumption of data center resources. Hu *et al.* [104] expressed problematic issue of VM placement as a multi-objective optimization problem. An improved ant colony algorithm is offered for the data centers to reduce total resource wastage and energy consumption. Liu *et al.* [105] gives ant colony optimization based solution for VM placement on physical

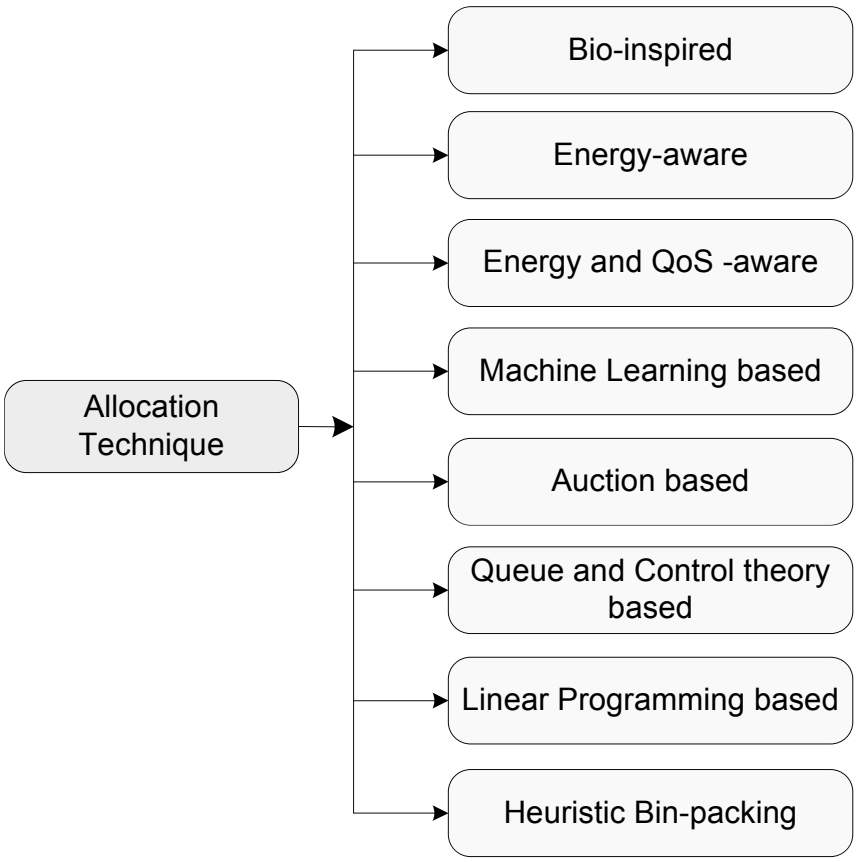


Figure 2.4: Taxonomy of allocation techniques

servers in order to decrease the number of active physical servers. Portaluri *et al.* [106] presented genetic algorithm based trade-off solution between tasks completion time and system power consumption. The system allocates resources to independent tasks on homogeneous single-core resources. Xiong and Xu [107] presented a multi-resource energy efficiency VM allocation model based on particle swarm optimization for energy efficiency of Cloud data center. Total euclidean distance is used as a fitness function to keep balance between resource utilization and energy consumption. This algorithm avoids falling into local optimal solution, which is common in traditional heuristic algorithms. Kumar and Raza [108] presented particle swarm optimization based strategy for VM allocation to PMs in order to reduce total energy consumption and resource wastage. Dashti and Rahmani [109] proposed modified particle swarm optimization solution to guarantee QoS to users' tasks, and reduce energy efficiency. Response time and deadline are considered as QoS parameters. This approach reallocate VM from the overloaded host, and dynamically consolidate under-loaded hosts for power saving.

Kansal and Chana [110] used firefly optimization to enhance energy efficiency of the Cloud without sacrificing performance. The authors used VM migration to enhance energy efficiency.

The comparison chart of the bio-inspired resource allocation techniques used in the Cloud is as shown in Table 2.1.

Huang *et al.* [113] proposed adaptive sub-optimal resource management scheme that uses Support Vector Regressions (SVRs) to estimate the number of resource utilization according to the Service Level Agreement (SLA) of each process. In the proposed scheme, global resource allocation module uses remaining resource table and resource utilization rate table to estimate number of VMs required to provide desired level of service. Genetic Algorithm (GA) is proposed for reallocation of resources to achieve better performance. The proposed technique suffers from single point failure. Moreover, centralized global resource allocation module, remaining resource table, and resource utilization rate table cause performance degradation when the number of requests are large. Gao *et al.* [111] proposed multi-objective ant colony system algorithm for virtual machine placement that minimizes total resource wastage and power consumption. The algorithm attempts to utilize server to its full capacity which would result in creation of hotspots and increase in number of SLA violations. Moreover, using server near full capacity causes more heat dissipation which results in decrease in server reliability.

Ant colony optimization technique for assigning real-time tasks to heterogeneous processors is proposed by Chen *et al.* [112]. Local search technique is applied to improve energy efficiency of the feasible assignment solution generated by the proposed assignment algorithm. The authors have claimed that their algorithm saves 15.8% energy over prototyped version of ant colony optimization.

2.5.2 Energy-Aware Resource Allocation

Beloglazov *et al.* [94] proposed power efficient and QoS aware resource allocation heuristics and an algorithm for minimization of number of VM migrations. Upper and lower threshold utilization levels are used to detect overloaded and underloaded machines. When the resource utilization of a particular server falls below the lower

Table 2.1: Bio-Inspired Metaheuristic Based Resource Allocation in Clouds

Authors	Metaheuristic	Obj. Func.	Infrastructure	Energy-Aware	Energy-saving	Technique	Platform
Gao *et al.* [111]	ACO	Multi-Objective	Single Machine	y	NA	simultaneously minimize total resource wastage and power consumption	Java language
Feller *et al.* [100]	ACO	Single	Homogeneous	Y	Consolidation	Multi-dimensional Consolidation to improve server utilization and decrease number of active machines	Implemented in Java
Chen *et al.* [112]	ACO	Multi-Objective	Single Machine	Y	NA	Deadline avoidance	Microsoft Visual C++ 7.1
Huang *et al.* [101]	GA	Single	Homogeneous	Y	NA	efficiency and optimization of resource allocation	CloudSim
Kansal and Chana [102]	ABC	Multi-objective	Homogeneous	Y	Consolidation	Improvement in Resource Utilization	CloudSim
Chimakurthi and Madhu Kumar [103]	ACO	Single	na	Y	NA	Adaptive Resource allocation with throughput and response time as QoS parameters	NA
Hu *et al.* [104]	ACO	Multi-objective	Homogeneous	Y	NA	VM placement for reducing total resource wastage	GridSim
Liu *et al.* [105]	ACO	Single	Homogeneous	Y	NA	VM placement to reduce number of active PMs	NA
Portaluri *et al.* [106]	GA	Single	Homogeneous	Y	NA	Power efficient resource allocation algorithm that optimizes task completion time and data center power consumption	Real testbed (Single Core PMs)
Xiong and Xu [107]	PSO	Multi-Ob	Homogeneous	Y	NA	energy efficient multiresource allocation model that improves resource utilization	CloudSim
Kumar and Raza [108]	PSO	Single	Homogeneous	Y	NA	Efficient VM allocation for resource wastage reduction	Simulation
Dashti and Rahmani [109]	PSO	Single	Homogeneous	Y	Consolidation	Reallocation of VMs for server consolidation, response time minimizatio and deadline avoidance	CloudSim
Kansal and Chana [110]	FFO	Multi-Objective	Homogeneous	Y	Consolidation	VM migration	CloudSim

ACO, Ant Colony Optimization; GA, Genetic Algorithm; ABC, Artificial Bee Colony: PSO, Particle Swarm Optimization; FFO, Fire Fly Optimization, CSO, Cuckoo Search Optimization

threshold value, all the VMs running on the machine are shifted to some other machine. If utilization of a machine is above upper threshold, one or more VMs are shifted to other machines to keep the utilization between the threshold values. They proposed algorithms for single core machines. Gao *et al.* [114] presented a dynamic resource management approach for energy saving and SLAs fulfillment. CPU speed is considered as the bottleneck of performance. DVFS and server consolidation are used for energy saving. Nathuji and Schwan [115] proposed energy efficient resource management architecture for virtualized data centers comprising local and global policies. Operating system's power management strategies are used in local resource management policies. Global policies handle consolidation of servers by applying live migration. Saurabh *et al.* [116] proposed Green Cloud computing framework for reducing carbon footprint without sacrificing QoS. The authors propose Green Offer Directory (GOD) and Carbon Emission Directory (CED) to offer green services to the users. The CED maintains data related to the energy efficiency of Cloud services. Based on the information in GOD and CED, the cost and carbon footprint of leasing a particular Cloud service are calculated. Gabriel *et al.* [117] proposed a model of hardware node by configuring four basic subsystems: computing, memory, storage, and network. Each node that represents a hardware device is linked to one energy meter module. The energy meter module is in charge of measuring the amount of energy consumed by attached hardware. Each energy meter is linked with energy manager. This module gathers the amount of energy consumed by each component in Cloud system. The energy manager module manages the amount of energy consumed by individual hardware component, aggregated consumption of each node, and the overall energy consumption. Kinger *et al.* [118] proposed event driven prediction based proactive temperature aware VM scheduling to keep temperature of a server below the specified upper threshold temperature. Temperature predictor periodically monitors temperature of the PM. The authors used unified list to store current as well as threshold temperature of each node which is updated after a fixed interval of time. Young [48] and AntonBeloglazov [119] proposed two energy-conscious consolidation heuristics in order to maximize resource utilization without affecting the performance of the system. Kim *et al.* [120] proposed a model for estimating energy consumption

of a VM based on its in-processor events without using dedicated energy measuring instrument. Performance counters available in modern processor are used to keep track of a specific floating point instructions issued by VM. Based on instruction type and its count, energy consumption of VM is estimated. Number of performance counters available in a processor is limited, so limited number of instructions can be tracked, which results erroneous estimation of energy consumption. The authors also proposed energy credit scheduler. The proposed scheduler assigns resources to the VM based on its energy credit. The resources allocated to VM are preempted when its energy credit vanishes. Due to inaccurate energy estimation, energy credit scheduler performance deviates. Wu *et al.* [121] presented a technique to increase the utilization and efficiency of hardware equipment. Dynamic voltage frequency scaling is employed to decrease energy consumption for executing jobs without sacrificing its performance. The varioud approaches to conserve energy and maximize resource utilization without affecting the performance of the system are given in [49, 115, 119, 122]. They used energy-conscious consolidation heuristics to improve utilization and energy efficiency.

Lee *et al.* [123] proposed performance analysis based resource allocation strategy for Green Cloud. Every PM in a data center is assigned a performance value based on CPU processing speed, number of core, and memory capacity relative to machine having maximum value of these parameters. PM is allocated to a VM if its performance value best fits VM requirements. The proposed method results in hotspots and overloading of high performance machines. Improper distribution of load among servers would cause wastage of energy. Raycroft *et al.* [124] analyzed the effect of global virtual machine allocation policy on energy consumption. Simulation is performed for same type of applications but real Cloud host diverse type of applications. Communication cost between VMs and QoS is not taken in to account. Moreover, the authors proposed movement of VMs between regions which is impractical in case of large sized VM. Chieu *et al.* [125] proposed an architecture for dynamic allocation of resources to workloads, based on threshold number of active sessions. The proposed work is capable of maintaining higher resource utilization, thus reducing infrastructure and management costs. Feller *et al.* [100] proposed multi-dimensional ant colony optimization based workload consolidation algorithm. The algorithm uses resource utilization history to predict future resource demands and dynamically overbooks the resources.

The authors have tested the algorithm on PMs having same capacity i.e. in homogeneous environment. However, the real Cloud environment is heterogeneous in nature, having machines with different resource capacity. In [117, 120], the authors proposed methods to calculate energy consumption of a PM.

Xu and Fortes [126] proposed multi-objective VM allocation algorithm. The authors have taken CPU, and memory parameters for VMs, and have claimed reduction in power consumption, thermal dissipation costs, and resource wastage. Disk utilization and inter VM communication cost is not taken into consideration. Takeda and Takemura [49] proposed ranking of physical servers for consolidation and VM placement. Servers with higher priorities are considered more reliable than the servers with lower priority value. Higher priorities are assigned to newly installed servers. The main drawback of server ranking strategy is that the priorities are to be assigned to a server by the operator manually. Yanggratoke *et al.* [257] also presented a decentralized solution for resource allocation to minimize energy consumption through VM consolidation. They used efficient heuristic based on gossip protocol for efficient resource allocations. Quan *et al.* [127] presented resource allocation framework that improves utilization, and hence the energy efficiency of the Cloud infrastructure. Comparison chart of various energy-efficient and QoS-aware resource allocation techniques is given in Table 2.2.

2.5.3 Energy and QoS Aware Resource Allocation

Srikantaiah *et al.* [128] have proposed a heuristic for workload consolidation over multi-dimensional resources, and investigated the effect of increase in utilization of resources due to consolidation on performance of the system. Wu *et al.* [129] advocated SLA based provisioning technique which reduces resource cost and SLA violations. The management of customer requests, mapping them with resources is defined along with the supervision of different types of workloads by considering execution time. Bobroff *et al.* [130] presented an algorithm for dynamic reallocation of VMs to PMs which pro-actively adapts to changes in demand and migrates VMs between PMs to provide SLA guarantees. Workload signatures are used to identify the VMs which benefit the most from dynamic migration and the gain that a VM achieves due to

Table 2.2: Energy and QoS-aware Resource Allocation in Clouds

Authors	Year	Energy	QoS	Energy-saving	Infrastructure	Adaptation	Objective Function	Workload	Platform
Beloglazov *et al.* [94]	2012	Y	Y	DVFS, and Consolidation	Heterogeneous	Reactive	NA	Heterogeneous	CloudSim
Gao *et al.* [114]	2014	Y	Y	DVFSand Consolidation	NA	Prediction	Multi-Objective	Heterogeneous	Read testbed
Nathuji and Schwan [115]	2007	Y	N	Consolidation	NA	NA	NA	NA	Read testbed
Saurabh et al. [116]	2011	Y	Y	NA	NA	NA	NA	NA	Simulator
Kinger et al. [118]	2013	Y	N	VM Migration	Homogeneous	Proactive	NA	Homogeneous	Read testbed
Lee and Zomaya [48]	2012	Y	N	Task consolidation	NA	NA	NA	Homogeneous	Simulator
AntonBeloglazov [119]	2012	Y	Y	NA	Single Machine	Proactive	NA	PlanetLab Workload	Clojure programming language
Kim et al. [120]	2014	Y	N	in-processor events	NA	Reactive	Single	Homogeneous	Xen virtualization system
Raycroft *et al.* [124]	2014	Y	N	Consolidation, turning off idle machines	Heterogeneous	NA	NA	Heterogeneous	Simulator
Xu and Fortes [126]	2010	Y	N	NA	NA	NA	Composite	Heterogeneous	Simulator
Wu *et al.* [121]	2014	Y	N	DVFS	Homogeneous	Reactive	NA	Homogeneous	CloudSim
[49]	2010	Y	N	Consolidation	Heterogeneous	NA	NA	Heterogeneous	Java
[122]	2011	Y	N	Consolidation	Homogeneous	Reactive	NA	Heterogeneous	Simulator
Lee *et al.* [123]	2013	Y	Y	Improving Utilization	Heterogeneous	Reactive	NA	Homogeneous	CloudSim
Chieu *et al.* [125]	2009	Y	N	NA	NA	Reactive	NA	Homogeneous	Not Mentioned
Quan *et al.* [127]	2014	Y	N	Utilization improvement	Heterogeneous	Prediction	NA	NA	Read testbed

QoS, Quality of Service

dynamic migration is also evaluated. Further, time series forecasting and bin packing heuristic is employed to minimize the number of PMs required. The algorithm is analyzed with traces from production data. Zhang *et al.* [131] put forth *Scattered*, a VM migration algorithm, targeting over-committed data centers to minimize: number of VM migrations, risk of overload and impact on communication cost among VMs. *Scattered* identifies VMs for migration, on the basis of correlation between workload of VMs and impact on VM's communication cost, upon identification of overloaded PM. They performed extensive simulation with real workload traces from production Cloud environment, but they considered only processor utilization of VMs. Foster *et al.* [132] presented a dynamic approach for VM management to satisfy dynamically changing management objectives of minimizing energy consumption and SLA violations. Their study addresses how to switch between these two goals through dynamically switching strategies. Yazir *et al.* [133] presented a distributed approach for VM migration wherein each physical machine has an autonomous node agent. The node agents use PROMETHEE method to collectively decide the configurations in parallel. The authors claimed that their approach is promising in terms of scalability, feasibility and flexibility. Nathani *et al.* [134] proposed modified immediate and advance reservation algorithms for deadline sensitive leases. The proposed algorithms try to schedule new lease as a deadline sensitive lease in a single or multiple time slots. If a new lease cannot be scheduled in single or multiple time slots, the algorithm reschedules the already scheduled deadline sensitive leases to make room for new lease. In case, rescheduling fails to generate deadline constrained schedule, then backfilling is applied to accommodate new lease. The drawback of proposed algorithm is its high lease preemption rate which, in turn, increases allocation overhead. Son *et al.* [135] introduced workload and location-aware resource allocation scheme *WLARA* with automated SLA negotiation mechanism but they did not consider energy efficiency.

Comparison chart of various energy-efficient and QoS-aware resource allocation techniques is given in Table 2.2.

2.5.4 Linear Programming Based Resource Allocation

Gao *et al.* [136] proposed linear programming based multi-objective ant colony based system for virtual machine placement to minimize resource wastage and power consumption. Initial pheromone value is assigned to VM-host movement. The pheromone value indicates probability of a host to be selected for allocation of VM under consideration. The authors used only CPU processing speed and memory requirements of VM while allocating resources to the VMs. Sharma *et al.* [137] put forth a cost aware provisioning approach for Cloud applications to minimize the total provisioning and reconfiguration cost of an application. They used replication and migration to dynamically provision capacity, and integer linear programming to optimize the cost. A prototype of provisioning engine is implemented on OpenNenula to evaluate the efficacy of proposed approach. The experiments demonstrated the cost benefits of the approach over prior cost-oblivious approaches and the advantage of combining both replication and migration-based provisioning into a single approach. Wei *et al.* [138] proposed a game-theoretic method for resource allocation of tasks with multiple dependent subtasks. The authors generated initial solution with binary integer programming method and then improved the initial solution with evolutionary algorithms to obtain a final optimized solution.

2.5.5 Auction Based Resource Allocation

Auction based resource allocation is a provisioning mechanism where the bidders specify their demand in terms of number and type of instances, and the maximum price they are willing to pay for it. The current price of a resource is determined from active bidders. The bidders having higher value than current price are allocated their desired instances. All the bidders pay the same price that is decided from auction. Comparison chart of auction based resource allocation techniques is given in Table 2.3.

Zhang *et al.* [139] presented a dynamic resource allocation mechanism to provision free data center capacity to spot market for revenue maximization. Time series analysis is

used to predict future demand of a spot market. The resource prices are dynamically fixed to maximize the expected revenue over time. The price of the resources is decided dynamically based on market conditions. Samimi *et al.* [140] proposed double sided auction based resource allocation model, which enables resource providers and consumers to participate in bidding on an unrestricted number of resources. The participating parties are motivated to reveal their valuation for resources. The approach is claimed to have economic efficiency and incentive capability.

Chard *et al.* [141] applied high performance economic resource utilization techniques to address some of the performance issues of economic resource allocation mechanisms such as slow reallocation rate and high overheads. The authors implemented quantified overbooking, advanced reservation, just-in-time bidding and second chance strategies, and have claimed increase in occupancy and obtainable utilization of Cloud datacenters. These techniques improve the performance of the auction based resource allocation system. The strategies are implemented in DRIVE [142] meta-scheduler and analyzed using synthetic workloads. Hu *et al.* [143] advocated a bidding-based method under the game theoretic framework for effective and secure allocation of resources in Cloud environment. The proposed bidding strategy obtains the Nash equilibrium solutions for multiple participants bidding for a group of resources. Further, a statistical mechanism is presented to detect the malicious contender from a group of non cooperative participants.

Wang *et al.* [144] presented a reverse auction based resource allocation that considers market efficiency as optimization criterion and also take fraudulent behavior of malicious participant into account. A Reverse Batch Matching Auction (RBMA) is introduced to improve the efficiency of resource allocation mechanism. Further, twice punishment technique is applied to detect the malicious users. Market efficiency, customer satisfaction and QoS are used for evaluation of resource allocation scheme. Ding *et al.* [145] advocated an adaptive strategy-proof and budget-balanced resource allocation strategy based on double auction, which ensures utility value for both service providers as well as consumers by trading resources at their expected price and quantity to ensure a utility value for them. The authors applied different strategies including over-provisioning, under-provisioning and market equilibrium in

Table 2.3: Auction based Resource Allocation in Clouds

Authors	Technique	Objective	Auction Method	Achievement	Platform
Samimi *et al.* [140]	Combinatorial Double Auction Resource Allocation (CDARA)	Economic efficiency and incentive capability	Combinatorial Double Auction	Bidding of unrestricted number of resources	CloudSim
Chard *et al.* [141]	Economic resource utilization techniques for high performance	Addresses slow reallocation rate and high overhead	Implemented and quantified overbooking, advanced reservation, just-in-time bidding, and second chance strategies	Increased occupancy and utilization of cloud data centers	Real testbed
Hu *et al.* [143]	Bidding-based method under the game theoretic framework	Effective and secure resource allocation of resource	Nash equilibrium solutions for multiple participants bidding	Bidding of a group of resources	C++ programming language and Matlab 7.0
Wang *et al.* [144]	Immune evolutionary algorithm is applied for optimal resource allocation	Market efficiency, user satisfaction and service quality	Reverse Batch Matching Auction	Twice punishment technique to detect malicious users	Simjava2.0
Ding *et al.* [145]	Budget-balanced resource allocation strategy	To improve Cloud user satisfaction degree and cloud resource utilization	Adaptive double auction	Different strategies including over-provisioning, under-provisioning and market equilibrium	NA
Sun *et al.* [146]	Nash equilibrium based cloud resource allocation algorithm	To achieve the maximum social utility	Continuous Double Auction	Performance-QoS and economic-QoS	CloudSim
Shi *et al.* [147]	Two stage hybrid bidding resource allocation policy	To Improve belief value for both the resource provider (seller) and resource consumer (buyer)	Continuous Double Auction	Competitive equilibrium price is calculated using recent transaction prices	NA
Zaman and Grosu [148]	Resource allocation mechanisms that supports auction of multiple instances of different types as a bundle	To improve resource utilization, generated revenue, and allocation efficiency Combinatorial Double Auction	Improve the allocation efficiency while generating higher revenue for the cloud providers. Simulated environment		
Özer and Özturan [149]	A penalty cost mechanism for better utilization of resources and increasing revenue depending on the application domain	Improved resource utilization	NA	Bidding on bundles of available resources	NA
Fujiwara *et al.* [150]	Double-sided combinatorial auction mechanism	Bidding of bundle of resources , maximize customer's total utility	Double-sided combinatorial auction	Linear programming is used for winners determination	W-Mart Simulator
Teng and Moguls [151]	A novel resource pricing and allocation policy	To satisfy budget and deadline constraints by predict the future resource price	Proportional share auction	Users can predict the future resource price, Satisfy budget and deadline constraints	CloudSim
Zhang *et al.* [152]	An incentive-Compatible (truthful) Online Cloud Auction mechanism for resource allocation	To satisfy heterogeneous demands of user	Truthful online auction	A monotonic payment rule and a utility-maximizing allocation rule, a bidding language is introduced	simulated

different scenarios to fulfill the requirements of larger number of participants. Sun *et al.* [146] proposed a resource allocation approach for meeting the performance and economic metrics of the Cloud computing consumers. It is based on game-theoretic continuous double auction strategy and incorporates Nash equilibrium to achieve the maximum social utility. Shi *et al.* [147] present resource allocation policy for Cloud that incorporates a continuous double auction. The competitive equilibrium price is calculated using recent transaction prices, which, in turn, is used for evaluating belief value for both the resource provider (seller) and resource consumer (buyer). Zaman and Grosu [148] presented two resource allocation mechanisms which are based on combinatorial double auction. The first mechanism employs linear programming and enables consumers to bid multiple instances of resources as a bundle. The second mechanism is based on greedy technique and decides the resource allocation on the basis of valuation of the customers, the total number of items and the relative size of the VM instances. The auctions are carried out periodically and each customer can bid once in an auction. The proposed mechanisms are incentive-compatible and guarantee that customers maximize their utility by bidding their true valuations for the requested bundles. The proposed mechanisms are extension of work by Archer *et al.* [153] and Lehmann *et al.* [154]. Unlike [153] and [154], it supports auction of multiple instances of different types as a bundle. The proposed mechanism is compared with two fixed price allocation mechanisms used in Windows Azure platform [155] and claimed to outperform them in terms of resource utilization, generated revenue, and allocation efficiency. Özer and Özturan [149] proposed a combinatorial auction based solution for resource co-allocation issue in both commercial and nonprofit Cloud environment. It increases the utilization of resources and provides fairness among customers. A penalty cost is associated with unallocated resources after an auction. Bidders can submit bids on bundles of available resources. It provides better allocation of resources to the bidders by reducing the number of unallocated resources to increase the overall utilization of the resources. Fujiwara *et al.* [150] presented resource allocation mechanism which is based on double-sided combinatorial auction mechanism and supports the bidding of bundle of resources to maximize customer's total utility. A bid describes the resources that constitute the bundle and the maximum price a customer is willing to pay for the bundle of resource. Linear programming is used for winners determination

among the bidders. Budget balance and individual rationality are taken care through pricing scheme. Teng and Moguls [151] proposed resource allocation mechanism with resource pricing for Cloud computing which is based on a proportional share auction. The bidders are allocated resources based on their bid proportions and can predict the future resource price to satisfy budget and deadline constraints. Tsai and Tsai [9] presented a resource allocation approach for multi-customer-multi-provider Cloud market which can adjust resource price adaptively. Zhang *et al.* [152] proposed a framework of truthful online auction for Cloud computing with heterogeneous customer demands. In this framework, Cloud customers can bid for Cloud resources in an online manner by a truth-telling scheme. Customers are classified into three valuation types to meet heterogeneous demands, each with a corresponding valuation function. Each customer translates its specific valuation into a concise request according to its own type. Customers valuation changes with respect to the allocation they get. The truthfulness is enforced by a non-decreasing auxiliary pricing function.

2.5.6 Queuing and Control Theory Based Resource Allocation

Kusic *et al.* [156] proposed performance and power efficient resource-management approach based on look-ahead control method for virtualized heterogeneous environments. Prediction is employed for dynamic reallocation of resources. Leva *et al.* [157] proposed a generic resource allocation approach that is based on a two-level time varying control scheme. They also analyzed and proved the stability of the proposed closed loop system. Patikirikorala *et al.* [158] made an appropriation of the multi-model nature of software systems. They proposed a framework called Multi-Model Switching and Tuning (MMST) for the performance management of software exhibiting multi-model behavior. Gandhi *et al.* [159] proposed an approach to minimize mean execution time of application in heterogeneous server farm within available power budget. The authors studied the impact of CPU frequency scaling on energy consumption, and observed linear power-to-frequency relationship. The authors used queueing model to predict the mean execution time as a function of power-to-frequency relationship, arrival rate, peak power budget, and so on. The experimental results against different

types of workloads showed that an efficient power allocation can significantly vary for different workloads.

2.5.7 Machine Learning Based Resource Allocation

Liu *et al.* [160] proposed SPRNT, a reinforcement learning-based aggressive resource allocation system for IaaS Cloud, to efficiently provision resources to rapidly changing workload. The reinforcement learning based engine continuously learn the relationship between resource allocation and performance of application. The SPRNT increases the workload allocations substantially in a single adjustment to avoid SLA violations. The allocated resources are later decreased slowly to reduce over-provisioning of resources if required. The efficiency of the proposed approach is evaluated with realistic workload traces of three web sites. It can achieve 7.7x speedup in adaptation time compared to existing efforts and restrict the SLA violations rate to 1.3% at the expense of 2.5-9.0% wastage of resource. Vasić *et al.* [161] proposed DejaVu, a system that simplifies and accelerates the management of virtualized resources in Cloud computing services. The previous resource allocation decisions are cached and utilized for efficient management of virtualized resources. When a change in workload condition is detected, cache is referred with VM identification and workload signature to decide new resource allocations. The workload clusters are build using machine learning to speedup cache lookup. DejaVu reduces resource management effort and overhead by limiting the number of resource re-configurations. Song *et al.* [162] presented an automated resource allocation system that can avoid overloading of servers whiling minimizing the number of servers in use. The concept of skewness is introduced to multiplex the different types of applications on a server in order to improve utilization. The authors also proposed load prediction algorithm which is used for efficient allocation of resources. Meng *et al.* [163] presented a VMs provisioning approach to allocate resources to group of VMs provisioned together on the basis of their aggregate resource needs. The approach exploits diverse workload patters of multiple VMs. The unused resource capacity of low utilized VMs are allocated to other co-locating VMs with high utilization. The authors evaluated the proposed method with production data collected from commercial data centers. Viswanathan *et al.* [164] proposed design and implementation of resource

allocation system (CloupMap) for private Clouds, which places an application with known usage pattern on a server hosting workload having complementary usage patterns. The effectiveness of the CloupMap is analyzed with production traces from live data centers. Koch *et al.* [165] presented a probalistic cost-optimization mechanism for Cloud which is based on "Maximum likelihood estimation". The authors aimed to improve the utilization of resources to cut-down allocation cost. Islam *et al.* [166] presented a prediction framework that exploited neural networks and linear regression along with sliding window based technique for on demand resource allocation in the Cloud. Mi *et al.* [167] applied Brown's quadratic exponential smoothing combined with genetic algorithm to perform resource prediction and auto-reconfiguration of VMs in Cloud data centers. Their approach switches idle PMs to sleep mode to reduce energy consumption.

2.5.8 Heuristic Bin-packing Based Resource Allocation

Whenever, a Cloud user requests for the provisioning of one or more VMs, the resource allocation system of the Cloud service provider discovers a suitable PM and allocates resources to VMs on it. The allocation decision should help achieve the provider's objectives. Generally, the VMs are assigned static shares of the PM's resources. The placement of VMs onto PMs can thus be considered as Vector Bin Packing (VBP) problem. Hence, VBP can be used to model resource allocation problems where the resources used by each PM are additive. Therefore, the optimal allocation is one where the VMs are packed into least number of PMs. The VBP and its variants are NP-hard problems [168], thus heuristic algorithms are commonly used in practice. Comparison chart of various heuristic bin-packing based resource allocation techniques is given in Table 2.5.

Wu *et al.* [121] proposed energy efficient priority job scheduling for Cloud computing. The requirements of a job are given in terms of maximum and minimum CPU frequencies. Every server is assigned some weight based on its performance/watt. Servers are selected for jobs according to assigned weight and SLA level required by the users. The job is assigned to VM running on selected server that meets its requirements. Frequency of server is then tuned to reduce energy consumption. However, authors

Table 2.4: Machine Learning based Resource Allocation in Clouds

Authors	Year	Technique	Learning Approach	Objective	Workload	Achievement	Platform
Liu et al. [160]	2014	Proposed an aggressive resource allocation system named SPRNT	reinforcement learning-based	Avoid SLOs violations	Realistic workload traces of three web sites	1.3% SLO violations at the expense of 2.5-9.0% wastage of resource, 7.7x speedup in adaptation time	KVM-based virtualization platform
Vasić et al. [161]	2012	Proposed DejaVu	Feature Subset Selection for Machine Learning	Minimizes the resource management overhead, quickly adapts to workload changes,deals with interference	NA	10x speedup in adaptation time	Amazon EC2
Song et al. [162]	2012	Skewness is introduced to multiplex the different types of applications	Load prediction using exponentially weighted moving average	Dynamic resource allocation as per application demand, Overload avoidance, and Green computing by optimizing the number of servers in use	NA	Improved utilization	Real Testbed
Meng et al. [163]	2010	Exploits statistical multiplexing among the workload patterns of multiple VMs	Prediction using ARMA and neural networks	Multiple VM provisioning at a time	Production data collected from commercial data centers	45% improvement in resource utilization	Simulation
Koch et al. [165]	2016	Probabilistic workload-aware dynamic resource allocation	Maximum likelihood estimation based	Improve the utilization of resources to cut allocation cost with minor impact on QoS	NA	NA	NA
Islam et al. [166]	2012	prediction-based resource measurement and provisioning strategies	Neural Network and Linear Regression based	NA	NA	NA	platform- Amazon EC2 cloud
Mi et al. [167]	2010	Auto-reconfiguration of virtual machines	Brown's quadratic exponential smoothing	Minimizing Energy Consumption	NA	NA	Real Testbed

SLO, Service Level Objectives; VM, Virtual Machine; ARMA, Auto Regressive Moving Average; EC2, Elastic Compute Cloud

have not considered memory, I/O and other requirements of the job. Panigrahy *et al.* [169] proposed Geometric Heuristic Algorithm (GHA) after experimenting and analyzing variants of the First Fit Decreasing (FFD) algorithm. GHA performance is at par with FFD for reasonable number of items and dimensionality. The authors claimed that the new heuristic outperforms FFD-based heuristics in most cases and sometime uses upto ten percent lesser number of bins. Li *et al.* [170], Jung *et al.* [171], and Gupta *et al.* [172] are some of the works which are based on bin packing heuristic and presented placement of VMs on PMs. Cardosa *et al.* [173] presented VM placement and power consolidation techniques for Cloud data centers which exploits min-max and share features inherent in virtualization tools such as VMWare to keep a balance between power consumption of infrastructure resources and performance of applications. They dynamically adjusted resource allocated to VMs encapsulating heterogeneous applications based on available resources, power costs, and application utilities to provide power-performance trade-off in modern data centers. Wolke *et al.* [174] analyzed simple bin packing algorithm for placement of VMs over PMs and convicted them for suboptimal due to ignorance of VMs which arrive in the future. Further, they conducted experiments and simulations with different workload environments and found that combinations of placement controllers and periodic reallocations achieve the highest energy efficiency. Bobroff *et al.* [130] presented an approach which periodically runs an offline bin packing algorithm to calculate VMs to PMs mapping. The approach eliminates hot-spots and minimizes the number of PMs in use.

Table 2.5: Heuristic Bin-packing based Resource Allocation in Clouds

Authors	*Year*	*Citations*	*Technique*	*Objective*	*EnergySaving*	*Platform*
Takeda and Takemura [49]	2010	29	Ranking of physical server for VM placement and consolidation	Energy	Consolidation	Simulated in java
Quarati *et al.* [64]	2013	20	Revenue, user satisfaction	Energy	Multiclass Single Queue to Multiple Servers	Simulator
Wu *et al.* [121]	2014	67	DVFS	Improves Utilication	CloudSim	SLA Aware
Panigrahy *et al.* [169]	2011	101	Geometric heuristic algorithm (GHA)	Minimize Number of PMs	–	–
Li *et al.* [170]	2009	245	EnaCloud - Dynamic VM placement	Live VM Migration	Energy Efficiency	iVIC platform
Jung *et al.* [171]	2008	43	Autonomic system proposed	Combination of bin packing and gradient search	Decision tree learning	RUBiS
Gupta *et al.* [172]	2008	67	Two stage heuristic algorithm	Reducing cost	Minimizing the number of target servers	Java
Cardosa *et al.* [173]	2009	251	Placement and power consolidation of VMs	Heterogeneous applications	Min-max resource share	Real Testbed
Wolke *et al.* [174]	2015	14	VMs placement	simulation	different workload environments	–

DVFS, Dynamic Voltage and Frequency Scaling; VM, Virtual Machine; SLA, Service Level Agreement; CPU, Central Processing Unit

2.6 Problem Formulation

It is apparent from the gaps perceived through the literature survey that energy aware resource allocation is a critical issue in Cloud computing environment. The problem of resource allocation is challenging due to large scale heterogeneous computing infrastructure and further division of each resource into multiple virtualized environments for co-hosting virtual servers. The immense energy use of these large scale data centers presents a great challenge to the researchers and the manufacturer. The principal reason of immense energy consumption is allocation of resources to the applications as per their peak demand. But due to dynamic nature of Cloud average resource utilization is approximately 15-20% [27,58]. Thus, a major proportion of total energy consumption is wasted due to over-provisioning of resources. The huge energy consumption not only lowers the profit share of service providers, but also poses an environmental threat due to massive carbon emissions [63]. Energy efficiency can be improved by increasing resource utilization. But over-committing of the physical resources decreases the performance of the applications and hence may cause SLA violations. Consequences of performance degradation can be critical. One direct consequence may be losing users. According to Hamilton [66], 100ms delay in reaction time decreases 1% sale on Amazon and 0.5s delay in search page generation decreases Google user traffic by 20%. This necessitates the development of novel resource allocation algorithms so that Cloud resources may be optimally allocated to users' applications. The resources should be allocated in such a manner that minimizes energy consumption while respecting QoS requirements of the end users. Further, the existing cloud providers such as Amazon Elastic Compute Cloud (EC2) offers customized operating system and applications in the form of virtual instances with heterogeneous resource dimensionality such as the number of processing cores, memory, and storage [175]. The virtualized instances are allocated required resources on the PM in First Come First Serve (FCFS) manner without considering other parameters of the encapsulated application such as QoS requirements, energy consumption, and budget constraints [176]. The awareness of an application's behavior and resources is necessary for optimal allocation of resources. The variable load of applications causes fluctuations in resource usage [176]. The resource requirements of an application can be predicted from its usage patterns such as

arrival rate, network usage, CPU usage and memory usage [177]. The usage patterns of the applications may be exploited to devise energy efficient and QoS aware resource allocation technique. Further, the Cloud consumers expect highly reliable and available Cloud services incurring minimum cost. Whereas, the Cloud service providers want to gain maximum profits by minimizing energy consumption through improved utilization of resources while respecting QoS requirements of the end user. The gaps analyzed in the literature survey necessitate development of a framework that provides energy efficient and QoS aware resource allocation for Clouds.
The following objectives are delineated for the proposed work:

(i). Explore and analyze the research in the area of resource allocation in Green Cloud to gain a systematic understanding of the existing techniques and approaches.

(ii). To propose energy efficient resource allocation algorithms in Green Clouds.

(iii). To evaluate and deploy the proposed algorithms on simulated/multi-node IaaS environment.

2.7 Summary

This chapter focused on exploring the existing literature for energy-aware resource allocation in Green Cloud computing. It presented Green Cloud computing, power management and QoS in Clouds. Further, the chapter analyzed the existing works on energy efficient resource allocation in Green Clouds. On the basis of literature survey, research problem has been formulated.
The next chapter presents a solution to the research problem by proposing a framework for Green Cloud for efficient and robust management of resources. The proposed framework addresses the issues identified in problem formulation and accomplishes the objectives of this research work.

Chapter 3

SERVmegh: A Green Cloud Framework [1]

The previous chapter discussed taxonomy and state of the art approaches in resource allocation and energy aware computing. It is analyzed from the existing literature that energy aware resource allocation has not been addressed to a large extent. To address the issue of huge energy consumption, a Green Cloud framework for efficient and robust management of resources has been proposed, and various components of this framework have also been implemented for energy efficient resource allocation in green clouds. This chapter starts with a brief discussion about different layers and their components. Then, it presents a comparison of the proposed framework with various open source as well as commercial Cloud environments followed by the derivation of mathematical expression for overhead of the layered architecture. Finally, it illustrates the resource allocation model, algorithms for energy efficiency, and performance evaluation of the proposed allocation approach on simulated as well as the real Cloud environment.

3.1 SERVmegh: A Framework for Green Cloud

The proliferation of Cloud computing and diversity in Cloud architectures demands for efficient management of resources and interoperability between clouds. The existing Cloud frameworks are not structured, i.e. applications deployed on one Cloud need to be modified for effective use of other Cloud capabilities. Also, scientific and resource-intensive applications demand for high performance computing infrastructures. These high performance large-scale data centers consume enormous amount of power. So,

[1] Ashok Kumar, Anju Sharma, Rajesh Kumar, *"SERVmegh: framework for green cloud"*, Concurrency and Computation: Practice and Experience, ISSN 1532-0634, 2016

there is a pressing need for an efficient yet scalable Cloud computing system framework. In this chapter, a Green Cloud computing framework for efficient and robust management of resources is proposed, that has been modeled in order to meet the goal of reducing power consumption in Green Cloud as shown in Figure 3.1. The framework is layered in nature. A layered framework divides functionalities across layers and also provides well defined interfaces for communication between layers. In general, there are two schemes for layering: strictly layered and relaxed layered. In the strictly layered scheme, functional units of a layer can interact with each other and the components of a layer that is immediate below this. However, in the relaxed layered scheme, a functional unit of a layer can interact with peers and components of any other layer.

In SERVmegh, relaxed layer design is used to support operational requirements such as maintainability, reusability, scalability, robustness, and security. The relaxed layered design improves efficiency because it does not demand strict forwarding of calls between layers. The relaxed layer design promotes reusability and helps to minimize the impact of changes in a component of a layer over the other layers. This feature allows to easily plug-in alternate implementations of a layer. In addition, an alternate implementation that returns a better result can be substituted for a layer that takes a long time to evaluate a solution. The client interact with the application layer of the SERVmegh which, in turn, uses the services of one or more layers. Each layer interact with other layers through well defined interfaces.

Benefits of relaxed layer architecture

- Maintenance and enhancements are easier to incorporate due to low coupling and high cohesion between layers.
- The functionality can be reused as the layer interfaces are designed with reuse in mind.
- Layered design support distributed development/deployment.
- Scalability, fault-tolerance, and performance improve if the layers are distributed over multiple physical tiers.

- Well-defined layer interfaces ease testing.
- Layered approach support switching out various implementations.

However, in relaxed layer approach changes in lower-level layer interfaces may percolate to higher levels. The effect of these can be reduced by limiting the boundary crossing of the layer.

The proposed framework comprises of six layers: *resource layer, virtualization layer, core service layer, security layer, management layer*, and *application layer*.
A brief description of all these layers and their components is as follows.

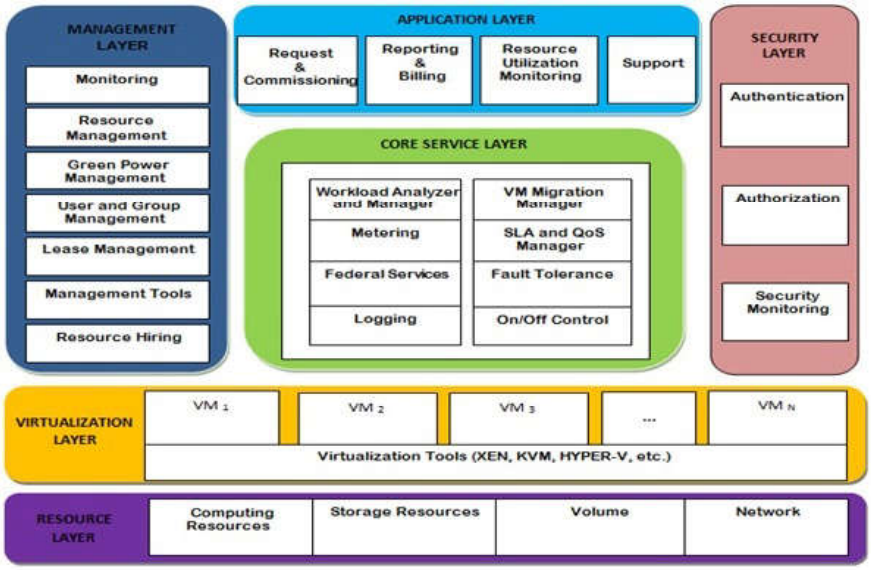


Figure 3.1: A Comprehensive Green Cloud Framework: SERVmegh

3.1.1 Resource Layer

This layer is a collection of all the manufacturing resources and capabilities including *computing resources, storage resources, volumes* and *network*. These resources are provided to users on 'pay per use' basis or rental basis. Service related details, like type of service to be provided, payments to be charged, etc. are thoroughly documented in SLA. The *computing resources* component takes care of all computational tasks, attaching/detaching volumes, acquiring console output, etc. This component is controlled by Cloud middleware that is designed and implemented to efficiently

provide, manage and schedule virtual as well as Physical Machines (PMs). *Storage resources* component provides mass storage facility to the users. Clients can store their data in this component. For instance, OpenNebula supports different types of data stores such as system, File-system, Internet Small Computer System Interface (iSCSI)/Logical Volume Management (LVM), Virtual Machine File System (VMFS), Ceph and Files [17]. Different storage types help in applying different SLA policies to different VMs, balancing Input/Output (I/O) operations between storage services. The *volume* component helps to manage block-level storage volumes that are persistent in nature. *Network* component plays a vital role in managing virtual network among a group of virtual machines. It assigns Media Access Control (MAC) and static IP addresses to VM instances, configures network among them and apply firewall to VMs.

3.1.2 Virtualization Layer

Virtualization is the idea of partitioning or dividing the resources of a single server into multiple segregated VMs. It enables users to have an illusion that the resources being used by a user are dedicatedly available to that user only and are not shared among users. In fact, resources are shared among a number of users simultaneously. It is an abstraction layer which partitions physical resources into dynamically scalable virtual resources that are provided to the users on demand. Virtualization of resources is possible because of the fact that, computing power of server machines is very high, and if such a machine is dedicatedly made available to a single user, then the enormous computing power of the server goes under-utilized. This leads to inefficiency in terms of energy. Virtualization can be accomplished by using virtualization tools such as Kernel-based Virtual Machine (KVM), XEN, and Hyper-V.

3.1.3 Management Layer

This layer provides features of *monitoring, resource management, green power management, user and group management, lease management, management tools,* and *resource hiring. Monitoring* component continuously monitors various activities that are going on Cloud. *Resource management* component manages physical as well as

virtual machines. It performs activities like starting, shutdown, reboot, etc. of virtual machines. With the help of *on/off control* manager of core service layer decides when to switch on/off a PM to conserve energy. *Green power management* component empowers Cloud administrators with the ability to apply different power management policies during peak/non-peak hours, and monitoring users' activities to predict future workload in order to provide uninterrupted service to the users. *User and group management* component helps in creating/deleting users/groups. Several users can be kept in a group to provide them same level of privilege. Different types of users like administrators, regular users and guest users can be created to provide them different levels of privileges. Only Cloud administrator has authority to create and destroy users or groups and assign privileges. *Lease management* controls leases resources to users, which can be provisioned to the user at some time in the future. It allows the users to request resources in a variety of terms, for example, "I need 6 nodes, each with 3 CPUs and 64 GB of memory, from 1 pm to 6 pm for three consecutive days, starting from tomorrow" [178]. *Management tools* component provides methods to interact with the lower layer services and resources of Cloud infrastructure. The method of interaction can be Command Line Interface (CLI), Cloud Application Programming Interface (API) or Graphical User Interface (GUI). Hosts and/or virtual networks can be added, deleted, or monitored using tools such as Virtual Network Computing (VNC) console, Euca2ools, OpenNebula CLI, etc. The user can allocate, deploy, migrate, suspend, resume, and shut down a virtual machine allocated to him at any time. He can access the Cloud services and resources through programming using APIs. *Resource hiring* component keeps track of resources hired from other Cloud providers.

3.1.4 Core Service Layer

Being an intermediary layer it plays a vital role as it connects the provider and the consumer. It contains *workload analyzer and manager, metering, logging, SLA and QoS manager, VM migration manager, fault-tolerance,* and *on/off control*. *Workload analyzer* keeps track of workload on servers in the Cloud. When the load on a particular server goes above upper threshold value, some of the tasks running on the node under consideration are migrated to other servers with the help of *VM migration manager*

to balance the load. *VM migration manager* manages live as well as cold migration of VMs. During VM migration, services become unavailable. Therefore, it is critically important to minimize the downtime. *Metering* component measures storage, compute, and other resources used by the consumer. Further, the usage metered by this component is used by billing and reporting unit to evaluate the cost to be charged. *Logging* component is the requirement of every Cloud system to meet its audit requirements. Every operation performed on Cloud even with administrator privileges is logged for security and performance purposes. The Cloud maintains a separate log file for different components/operations of the Cloud. For instance, check-pointing information logged in a file named *checkpoint*. *SLA and QoS* component keeps track of formally defined service contract, guarantee a certain level of performance, and provide different priority to applications or users. In case of violations of SLA, the Cloud provider pays penalty for percentage of clients' requests that do not meet specified upper bound on their service time. SLA enforcement ensures that QoS parameters are fulfilled with minimum SLA violations. *Fault tolerance* component predicts occurrence of fault and takes corrective actions for successful execution of the compute intensive applications. *On/Off* component switches a server to sleep mode when it's utilization falls below preset lower threshold value. The *on/off control, workload analyzer and manager*, and *VM migration manager* of *core service layer* are designed, implemented, and tested in CloudSim [176].

3.1.5 Security Layer

Major components of this layer are *authentication, authorization and security monitoring*. *Authentication* component allows a user to access Cloud services through username-password, Secure Shell (SSH) login or Kerberos time stamped Ticket-Granting Ticket (TGT) authentication method. The authenticated users are allowed to schedule tasks on the deployed VM based on their identity. For example, only administrators can create or delete virtual network while a normal user cannot perform these tasks. *Security monitoring* component detects malicious activity or procedural errors over the network. It can identify the attacks originating from inside/outside of the network. It may also analyze events in order to identify the attacks.

3.1.6 Application Layer

The layer is geared to the needs of all users in the Cloud environment. All the members of Cloud system, in the scope of their authority, can make a request any time and anywhere through their convenient ways. This layer mainly provides a service interface to service requesters in certain organized form. It can be a common interface like a web portal, or special interface meeting the request of the specific manufacturing application individual, industry or field. *Request and commissioning* component accepts user's requests for commissioning new virtual machines. *Billing and reporting* component generates bills for the amount of resources used by the client. It sends billing details to the users and processes payments, track invoices, etc. *Resource utilization monitoring* component continuously monitors usage of various resources that are provided as a service. *Support* component helps users in case of any problem in using, commissioning, billing, etc.

The comparison of the proposed Green Cloud framework, SERVmegh, with various open source as well as commercial clouds, including Open Nebula, Eucalyptus, Open Stack, Nimbus, Amazon web services is presented in Table 3.1.
The layered architecture bears some overhead in terms of communication cost. The next section derives expression for the overhead borne by the proposed layered architecture.

3.1.7 Layered Architecture Overhead

The proposed framework follows layered architecture design. Each layer providing a bundle of services through well defined interfaces can be deployed on separate PM or some of the layers could be deployed on same PM. The deployment of layers on various PMs requires physical connectors between them for exchange of information. The connectors introduce overhead in the form of communication cost. The overhead depends on the communication between various layers of the framework.
The proposed framework is modeled as Discrete Time Markov Chain (DTMC) [179]. It is represented by $n+2$ states set, $\delta = \{s_0, s_1, ..., s_{n+1}\}$, as shown in Figure 3.2.

Table 3.1: Comparative Analysis of different Clouds

Layers and Components	Open Nebula	Eucalyptus	OpenStack	AWS	Nimbus	MS Private Cloud	SERVmegh
Resource Layer							
Compute	Y	Y	Y	Y	Y	Y	Y
Storage	Y	Y	Y	Y	Y	Y	Y
Volume	Y	Y	Y	Y	Y	Y	Y
Network	Y	Y	Y	Y	Y	Y	Y
Core Service Layer							
Workload Analyzer	Y	Y	Y	Y	Y	Y	Y
Metering	N	N	N	N	Y	Y	Y
Migration Manager	N	N	N	N	N	Y	Y
SLA and QoS Manager	Y	Y	Y	Y	Y	Y	Y
Fault tolerance	N	N	N	N	Y	Y	Y
Logging	Y	Y	Y	Y	Y	Y	Y
On/Off Control	Y	Y	Y	Y	Y	Y	Y
Security Layer							
Authentication	Y	Y	Y	Y	Y	Y	Y
Authorization	Y	Y	Y	Y	Y	Y	Y
Monitoring	Y	Y	Y	Y	Y	Y	Y
Application Layer							
Request and Commissioning	Y	Y	Y	Y	Y	Y	Y
Reporting and Billing	Y	Y	Y	Y	Y	Y	Y
Resource Utilization Monitoring	Y	Y	Y	Y	Y	Y	Y
Support	Y	Y	Y	Y	Y	Y	Y
Management Layer							
Monitoring	Y	Y	Y	Y	Y	Y	Y
Resource management	Y	Y	Y	Y	Y	Y	Y
Green Power Management	N	N	N	N	N	N	Y
User and Group Management	Y	Y	Y	Y	Y	Y	Y
Lease Management	N	N	N	N	Y	Y	Y
Management Tools	Y	Y	Y	Y	Y	Y	Y
Resource Hiring	N	N	N	N	Y	Y	Y

The state s_0 denotes end user and state s_{n+1} corresponds to completion of requested service. Each state from s_1 through s_n of DTMC corresponds either to a layer or a connector. For brevity, transitions between states are not shown.

Transitions between the states of DTMC represent the transfer of control from one layer to another characterized by transition probability matrix P of size $(n+2)*(n+2)$. Each element p_{ij} of matrix P represents the probability of transition from state s_i to state s_j. End user submits a service request to the application layer, i.e. state s_1, with transition probability 1. The end user's request for a service may be processed

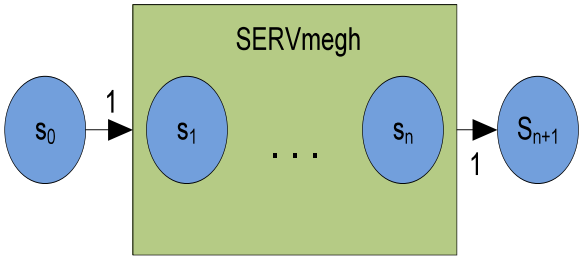


Figure 3.2: Discrete Time Markov Chain for SERVmegh

to completion by one or more layers. Upon completion, a service request transitions to state s_{n+1} with probability 1. The transition probability matrix is affected by the operational profile [180] and it is evaluated offline from the log file that keeps record of exchange of packets between layers.

Some computation and I/O are performed for each service request as the control passes through the layers. The computation and I/O may be performed repeatedly till the layer has completed its task. When a layer has finished with its part of the requested service, then it either returns the result, or passes the request to the next layer. The expected number of visits V_i to state s_i can be evaluated by solving the following system of linear equations [181].

$$V_i = q_i + \sum_k p_{ki} V_k, \quad i = 1, 2, ..., n \tag{3.1}$$

where q_i is the probability that service request starts at layer i. Once the visit count for all the states are calculated, the total service requirements on actual CPU and disks on PMs as well as connectors can be evaluated.

Total CPU and disk requirements of all the layers can be calculated from equations (3.2) and (3.3).

$$TR^p = \sum_{j \in PM_L} \sum_{i \in L_j} V_i * CPU_i, \tag{3.2}$$

$$TR^d = \sum_{j \in PM_L} \sum_{i \in L_j} V_i * Disk_i, \tag{3.3}$$

where, TR^p and TR^d are total CPU and disk requirements, respectively. CPU_i and $Disk_i$ are average CPU time and disk time per service request for layer i on PM j.

PM_L is a set of PMs on which all the layers are deployed, and L_j is set of layers collocated on j^{th} PM.

Total processing cost T^p can be evaluated from:

$$T^p = \zeta * TR^p + \xi * TR^d, \tag{3.4}$$

where, ζ and ξ are CPU and disk costs per unit time.

Total service requirements for all the connectors can be evaluated from:

$$TR^c = \sum_{i \in \kappa} packet_i / cap_i * V_i, \tag{3.5}$$

where TR^c is total service requirement of all the connectors, $packet_i$ is packet size transmitted by connector i, and cap_i is communication capacity of i^{th} connector. κ is a set of all the connectors.

Total cost of communication between layers can be evaluated from equation (3.6).

$$T^c = \gamma * TR^c, \tag{3.6}$$

where T^c is total cost of communication, and γ is communication cost in mega byte per second.

Overhead cost of layered architecture, T^o, can thus be calculated from:

$$T^o = \frac{T^c}{T^p}. \tag{3.7}$$

In the next section, Resource Wastage Reduction (RWR) methodology used for energy efficient resource allocation in SERVmegh is elaborated. Energy efficient resource management algorithms for RWR, VM placement, and minimization of VM migrations are also presented.

3.2 Resource Allocation Model

The resource allocation model based on resource wastage reduction methodology is as shown in Figure 3.3. A server/PM is modeled to have processor, memory, and network bandwidth. The *workload analyzer* evaluates the utilization of the server in terms of processing speed, memory, and network bandwidth. *Green Power Management* module composed of *Resource Utilization* (RU) and *Wastage Reduction Manager* (WRM). RU component stores utilization information of CPU, memory, and network bandwidth. WRM evaluates the resource capacity of the server that is being wasted. For example, suppose two VMs are running on a server with CPU, memory, and net-

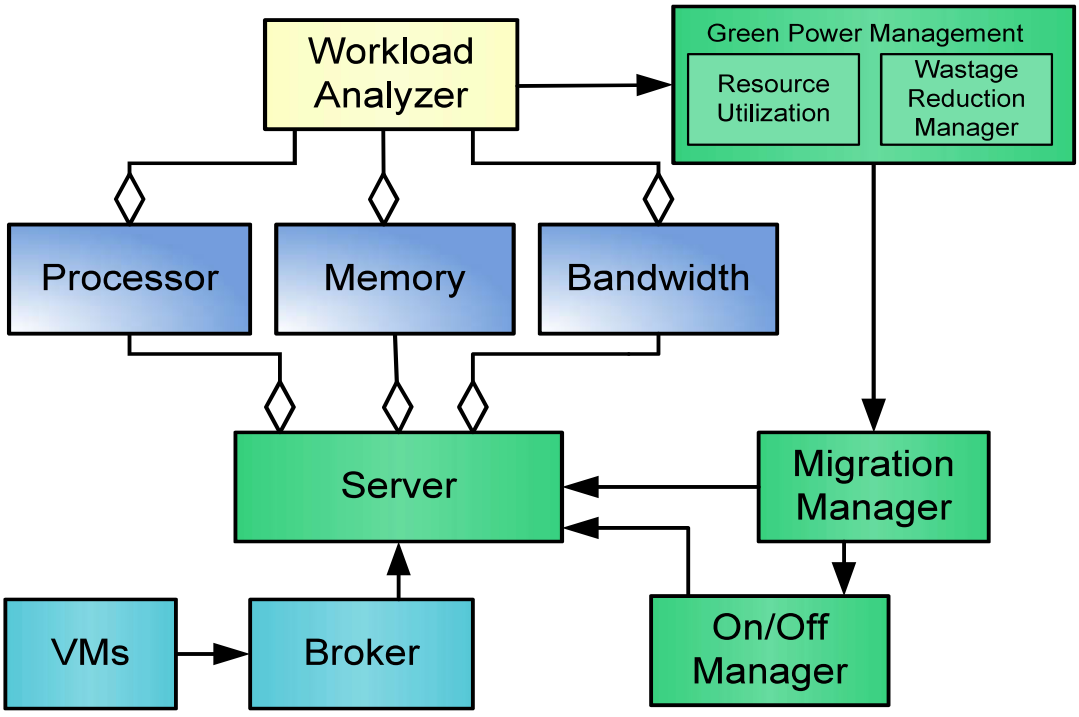


Figure 3.3: Usecase diagram for Core Service Layer

work bandwidth demand in the form of requirement vector $< 40\%, 30\%, 20\% >$ and $< 60\%, 55\%, 35\% >$. So the total requirement of both the VMs is $< 100\%, 85\%, 55\% >$. Two VMs are using the full capacity of the processor, but 15% of memory capacity and 45% of network bandwidth are being wasted. WRM balances the load among active set of servers. Based on the value of WRF, it finds the machines which are underloaded or overloaded. Overloaded PMs cause SLA violations. If a PM is judged

overloaded then some of VMs executing on it are migrated to some other PM(s). Selection of VM(s) for migration is taken care of by *migration manager* discussed in the later part of this section. *Migration manager* module selects VMs for migration. A VM that causes least change in the wastage reduction factor of the server is selected for migration. If a server is judged underloaded, all the VMs running on it are shifted to other server(s) and then it is switched to sleep mode by the *on/off manager*. Broker uses some technique to map VMs to PMs.

3.3 Resource Wastage Reduction Methodology

It is assumed that the utilization of system fluctuates as the system is subjected to variable load, workload can be unevenly distributed among the VMs, and workload can also be migrated from one server to another in order to balance load among servers. Suppose N VMs are to be placed on M servers. The requests from clients are expressed as VMs which are preloaded with required software. The resource requirements of a VM are expressed as 3-dimensional vector. Each dimension corresponds to specific types of the requested resource like processing speed (ps), memory (mem), and network bandwidth (nbw). The resource capacity of i^{th} server is represented by capacity vector $C_i = [C_i^{ps}, C_i^{mem}, C_i^{nbw}]$, i=1, 2, 3, ..., M. Where C_i^{ps} is CPU processing speed, C_i^{mem} is memory, and C_i^{nbw} is network bandwidth capacity of i^{th} server.

In virtualization environment more than one VM can be scheduled on a server and each VM has some resource requirements in terms of processing speed, memory, and network bandwidth. The resource requirements of j^{th} VM are represented by request vector $r_j = [r_j^{ps}, r_j^{mem}, r_j^{nbw}]$, j=1, 2, 3, ..., N. Where, r_j^{ps} is processing speed, r_j^{mem} is memory, and r_j^{nbw} is a network bandwidth requirement. When a VM is launched on a server, it uses the resources of the server and hence increases utilization and power consumption of the server. An idle server consumes 50% of the power consumed at full load [59]. Power consumption of servers increases linearly with increase in utilization. Utilization of server i is depicted by utilization vector $U_i = [U_i^{ps}, U_i^{mem}, U_i^{nbw}]$, i=1, 2, 3, ..., M. Where U_i^{ps} is the processing speed, U_i^{mem} is memory, and U_i^{nbw} is network

bandwidth utilization. Processing speed, memory, and network bandwidth utilization of i^{th} server can be calculated from equations (3.8), (3.9), and (3.10), respectively.

$$U_i^{ps} = \sum_j r_j^{ps} * x_{ij}, \quad \forall\ i, \tag{3.8}$$

$$U_i^{mem} = \sum_j r_j^{mem} * x_{ij}, \quad \forall\ i, \tag{3.9}$$

$$U_i^{nbw} = \sum_j r_j^{nbw} * x_{ij}, \quad \forall\ i, \tag{3.10}$$

where x_{ij} is 1 when j^{th} VM is instantiated on i^{th} server, 0 otherwise. The unused resource capacity of i^{th} server which is represented by $R_i = [R_i^{ps}, R_i^{mem}, R_i^{nbw}]$ can be evaluated from equation (3.11).

$$R_i = C_i - U_i, \quad \forall\ i, \tag{3.11}$$

where R_i^{ps}, R_i^{mem}, and R_i^{nbw} are the unused processing power, memory, and network bandwidth of i^{th} server, respectively. The unutilized resources on each server should be balanced along all the three dimensions. The unbalanced residual resources of a server prevent any further VM placement and thus result in wastage of computing resources.

3.4 Objective Function

The proposed RWR approach improves energy efficiency by reducing resource wastage along all the three dimensions, i.e. processing speed, memory, and network bandwidth. RWR uses Normalized Resource Wastage Vector (NRWV) to curb resource wastage. NRWV of server i is given by $\omega_i = R_i/C_i$. The high value of ω_i indicates an inefficient use of the resources. The lower is the value of ω_i the better is the utilization of resources. When the server utilization is 100%, value of ω_i becomes 0. ω_i value below the lower threshold value T_w^{lower} causes performance degradation.

A VM is mapped onto a server that has maximum value of Wastage Reduction Factor

(WRF), $\Re$. WRF can be evaluated from equation (3.12).

$$\Re_i = (\omega_i^t - \omega_i^{t-1})/\omega_i^{t-1}, \quad \forall\ i, \tag{3.12}$$

where ω_i^{t-1} and ω_i^t is NRWV of server i at time $t-1$ and t, respectively.

The algorithms for resource allocation based on resource wastage reduction methodology are presented in the next section.

3.5 Allocation Algorithm based on Resource Wastage Reduction Methodology

For RWR methodology three algorithms, namely: energy efficient resource utilization using RWR methodology, VM placement algorithm, and an algorithm for minimum number of VM migrations (MinMigration) have been proposed and implemented. Pseudo code for RWR methodology that enhances energy efficiency of servers by improving their resource utilization is presented in Algorithm 3.1. An array x of size $M * N$ is used to keep a record of VMs running over a PM. For instance, if a VM j is running over server i, then $x_{ij} = 1$, and 0 otherwise. NRWV and WRF are evaluated for each server pm_i. Positive value of WRF signifies improvement in resource utilization and hence increases energy efficiency of the server. NRWV value is tested to check for any scope of further improvement in energy efficiency. If NRWV value is less than the lower threshold then some of the VMs are shifted to other servers to keep resource utilization within the specified threshold values. A VM is selected for migration which causes least change in the value of $\Re$ of the server. In case, NRWV value is more than upper threshold, VMs running on the server are migrated to PM(s) that have maximum resource wastage and the server is switched to sleep mode to save energy. In line 13, if statement tests whether any VM is running on server i or not. Use of $VM(i)$ in the same statement is to get identification of all the VMs running over ith server. Algorithm 3.2 is for VM placement. For VMs to PMs mapping, the VMs are arranged in the decreasing order of the resource request vector and the PMs are arranged in the

Algorithm 3.1 Pseudocode for energy efficient resource utilization using resource wastage reduction

1: **Input:** VMs to PMs map, Set S of PMs
2: **Output:** Modified VMs to PMs map
3: **for all** $pm_i \in S$ **do**
4: Get VMs running on pm_i.
5: Calculate utilization vector U_i for pm_i from equations (3.8), (3.9), and (3.10).
6: Evaluate NRWV ω_i for pm_i
7: Evaluate $\Re_i$ for pm_i from equation (3.12)
8: **if** $\Re_i < 0$ and $\omega_i > T_w^{upper}$ **then**
9: $mwNode = MaxWastageNode()$
10: Select a VM_s for migration which results in minimum decrease in value of $\Re_i$
11: Migrate VM_s to $mwNode$
12: Set $x_{i,mwNode} = 1$ and $x_{i,s} = 0$
13: **if** $\bigvee_{j \in VM(i) \wedge j \neq s} x_{ij} = 0$ **then**
14: Switch pm_i to sleep mode
15: **end if**
16: **end if**
17: **if** $\Re_i > 0$ and $\omega_i < T_w^{lower}$ **then**
18: $mwNode = MaxWastageNode()$
19: Select a VM_s for migration which results in minimum decrease in value of $\Re_i$
20: Migrate VM_s to $mwNode$
21: Set $x_{i,mwNode} = 1$ and $x_{i,s} = 0$
22: **if** $\bigvee_{j \in VM(i) \wedge j \neq s} x_{ij} = 0$ **then**
23: Switch pm_i to sleep mode
24: **end if**
25: **end if**
26: **end for**
27: **return** Modified VMs to PMs map

decreasing order of capacity. Lines 8-16 find a server with a minimum value of WRF for every VM_j. In step 10, evalWRF method is called to find the wastage reduction factor of i^{th} server for VM_j with request vector r_j. Lines 17-21 assign VM_j to a server that not only fulfills the resource demands of the VM but also causes least resource wastage. The proposed algorithm reduces resource wastage to leverage the resource capacity of servers efficiently and hence reduces energy consumption.

Algorithm 3.3 selects VM(s) for migration. The elements of the power set of VMs running on the overloaded physical machine are arranged in increasing order of processing speed requirements. Lines 8-14, finds a set from the power set which results in the least change in the value of $\Re$, and whose total processing speed requirement is

Algorithm 3.2 Pseudocode for VM Placement based on resource wastage reduction methodology

Input: Set of VMs, Set of PMs
Output: VMs to PMs map
Arrange VMs in decreasing order of request vector
Arrange servers in decreasing order of capacity vector
for $j = 1$ to N **do**
  $minWRF \leftarrow 0$
  $machineID \leftarrow -1$
  for $i = 1$ to M **do**
    if $r_j < R_i$ **then**
      $WRF \leftarrow evalWRF(i, j)$
      if $minWRF > WRF$ **then**
        $minWRF \leftarrow WRF$
        $machineID \leftarrow i$
      end if
    end if
  end for
  if $machineID \neq -1$ **then**
    allocate VM_j to machine having machineID
  else
    Power on a new machine and allocate VM_i to it.
  end if
end for
return VMs to PMs map

Algorithm 3.3 MinMigration (PM)-pseudo code to find VMs to be migrated from overloaded PM

Input: Set PM of overloaded physical machines
Output: Set of virtual machines, VM_m, to be migrated.
vmList ← PM.getVmList()
vmPowerSet ← vmList.powerSet()
Arrange elements of $vmPowSet$ in increasing order of their total processing speed requirement
$overusage = PM.U_i^{ps} - T_{cpu}^{upper}$
$len = vmPowSet.length()$
for $k = 1$ to len **do**
  assign k^{th} element of vmPowSet to vmSet
  if $overusage < vmSet.totalCpuUtil()$ **then**
    $VM_m \leftarrow vmSet$
    goto step 15
  end if
end for
return VM_m

just greater than the difference between processing speed being utilized and the upper threshold value. The selected migratable set of virtual machines, VM_m, is returned as output.

After the discussion of the resource allocation methodology, objective function, and algorithms based on RWR methodology, the following section presents performance evaluation of the proposed approach on simulated as well as Open Nebula based private Cloud environment.

3.6 Experimental Results

The performance of RWR is evaluated in real as well as the simulated Cloud environment. Real Cloud environment is setup on forty PMs. Each physical machine is configured with Ubuntu 14.04 and KVM virtualization [182]. However, the simulated Cloud environment is created using CloudSim [176] simulator. CloudSim supports simulation of various Cloud entities such as data center, physical machines, virtual machines, etc.

3.6.1 Performance Evaluation in Simulated Cloud Environment

For evaluation of RWR methodology, two parameters i.e. performance and scalability are taken into consideration. Wastage reduction algorithm is rigorously tested on CloudSim by varying number of PMs and VMs.

The specifications of four types of servers simulated to build datacenter are as given in Table 3.2. These servers are simulated using *PowerHost* class of CloudSim and their corresponding power models are used to evaluate energy consumption. The VM request vector is varied along the 3-dimensions, i.e. processing speed in millions of instructions per second (mips), memory in megabyte (MB) and network bandwidth in megabits per second (Mbps). r_j^{ps}, r_j^{mem}, and r_j^{nbw} of VM requirement vector can take any value from set { 250, 500, 1000, 1500, 2000, 2500, 3000}, { 250, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000}, and { 250, 500, 750, 1000}, respectively.

Table 3.2: Specification of Physical Machines

Type	*CPU*	*Cores*	*RAM*	*Storage*	*BW*
1	2933	4	8	500	10
2	3067	4	8	500	10
3	2933	12	12	500	10
4	3067	12	16	500	10

CPU, processing speed in mips; Cores, number of processing cores; RAM, random access memory in GB; Storage, Permanent storage capacity in GB; BW, network bandwidth in Gbps.

3.6.1.1 Performance

For performance evaluation, RWR algorithm is compared with two bin packing algorithms, i.e. First Fit Decreasing (FFD), and Best Fit Decreasing (BFD). FFD algorithm allocates each VM in a list that is sorted in decreasing order of VM size to the first server which fulfills its resource requirements. However, BFD algorithm allocates each VM in a list that is sorted in decreasing order of VM size to a server which best fits its resource requirements. FFD and BFD are used for comparative analysis because these are similar to Packing policy of OpenNebula [183] and Greedy approach of Eucalyptus [184].

For performance analysis, three scenarios are taken into considerations. In each scenario, simulation is repeated 25 times and the number of VMs are varied from 200 to 1000. Requirement vectors of VMs are sorted in decreasing order by one of the 3-dimensions, i.e. processing speed, memory, or network bandwidth. The first scenario is performed with a list of VMs sorted on processing speed and the algorithms under consideration are named as FFD-PS, BFD-PS, and RWR-PS. In the second scenario, VMs are sorted according to memory requirement and the techniques under consideration are named as FFD-MEM, BFD-MEM, and RWR-MEM. Similarly, in the third scenario VMs are sorted according to NBW and the algorithms are named as FFD-NBW, BFD-NBW, and RWR-NBW.

Figure 3.4 shows a comparison of RWR with FFD and BFD in terms of resource wastage. Resource wastage is shown along the 3-dimensions, i.e. processing speed

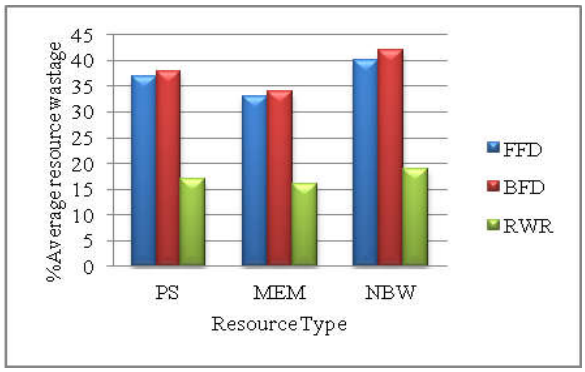


Figure 3.4: Resource wastage comparison based on Algorithm 3.1

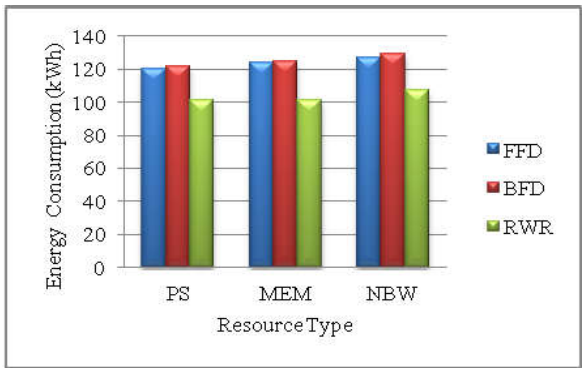


Figure 3.5: Energy Consumption Comparison based on Algorithm 3.1

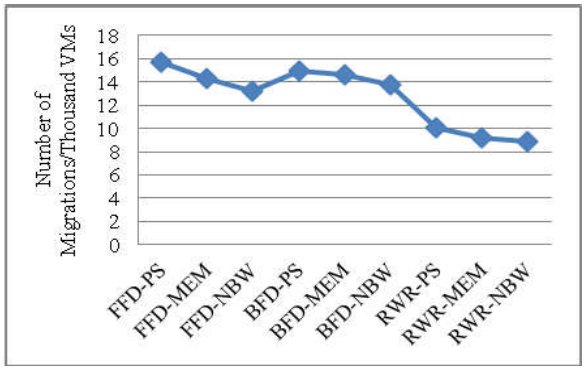


Figure 3.6: Comparison of number of VM Migrations

(PS), memory (MEM), and network bandwidth (NBW). PS wastage is 17% for RWR, whereas it is approximately 31% for FFD and BFD. MEM wastage is 16% for RWR and approximately 33% for both FFD and BFD. However, NBW wastage is 19%, 40% and 42% for RWR, FFD, and BFD, respectively. Resource wastage for RWR is least in all the three cases. Low resource wastage improves energy efficiency. The results indicate that RWR reduces resource wastage and hence increases energy efficiency of the Cloud infrastructure.

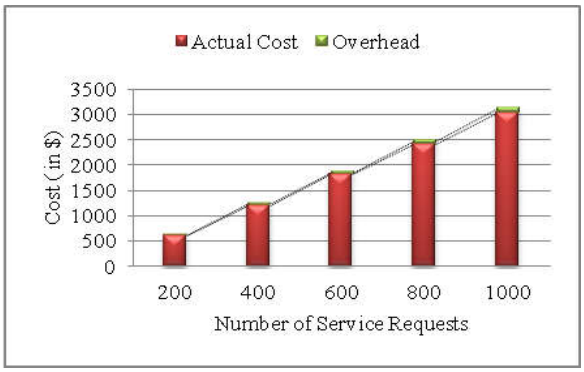


Figure 3.7: Overhead of Layered Architecture

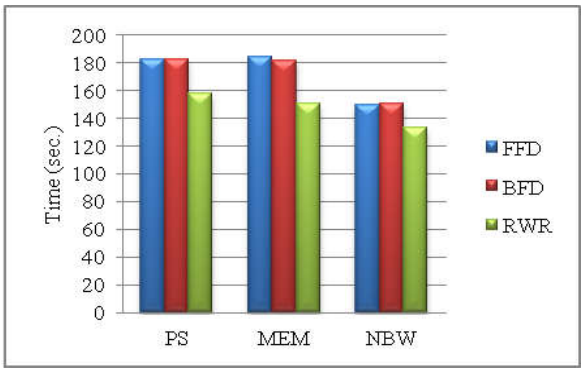


Figure 3.8: Comparison of VM Placement Performance (1000 VMs)

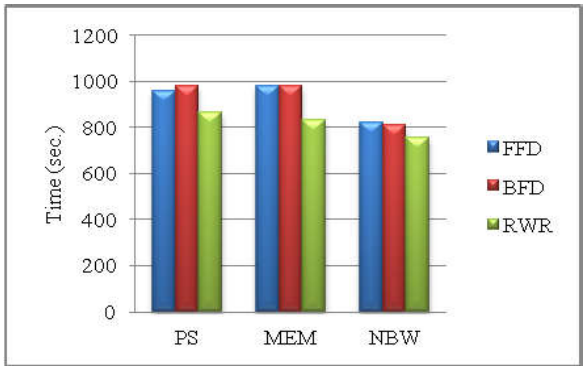


Figure 3.9: Comparison of VM Placement Performance (5000 VMs)

Figure 3.5 shows an energy consumption comparison of FFD, BFD, and RWR for three scenarios. In the first scenario, VMs are sorted in decreasing order of PS requirement. In this case, energy consumption of RWR is 15.83% and 17.21% lesser than FFD and BFD, respectively. In the second scenario, VMs are sorted in decreasing order of MEM requirements and energy consumption of RWR is measured 18.54% and 19.20% lesser than FFD and BFD. However, 15.74% and 17.05% lesser energy consumption is observed for RWR than FFD and BFD in the third scenario, for which the VMs

are sorted in decreasing order of NBW requirement. Energy consumption is minimum in case of RWR, due to the fact that RWR maintains utilization of servers between energy efficient utilization bounds. Moreover, RWR reduces energy consumption further by switching a server to sleep mode when its wastage increases beyond the upper threshold wastage level W^{upper}.

Figure 3.6 shows a comparison of number of VM migrations performed in the FFD, BFD, and RWR. In each simulation run for VM migration analysis, 1000 VMs and 200 PMs are used. We used 50 PMs of each PM type given in Table 3.2. VM migration causes unavailability of services and may thus cause SLA violations. In RWR approximately 30% lesser number of VM migrations are observed than FFD and BFD. It is evident that the number of migrations in RWR-PS, RWR-MEM, and RWR-NBW are least. Therefore, RWR provides better QoS and is more energy efficient than FFD and BFD.

Figure 3.7 shows the overhead of a layered architecture. The number of service requests is varied from 200 to 1000. A service request is handled by a VM. Per unit cost of CPU, disk, and communication are assumed to be \$0.084, \$0.042, and \$0.021, respectively. Probability of transition between states, average CPU time, and average disk time are evaluated offline from log file. Overhead cost of 2.95% is observed for 200 service requests which increases slightly with the increase in number of service requests and becomes 3.12% for 1000 service requests.

3.6.1.2 Scalability

In order to test the scalability of the proposed algorithms, large sized data centers with high VM requests are created in CloudSim. Two scenarios have been taken into consideration for mapping of VMs to PMs. In the first scenario, 1000 VMs and 400 servers are used. Whereas, in second scenario 5000 VMs and 2000 PMs are used. Results of the two scenarios are shown in Figure 3.8 and Figure 3.9 respectively. It is clear from both the figures that RWR takes less time to map a large number of VMs to physical machines, hence it is scalable.

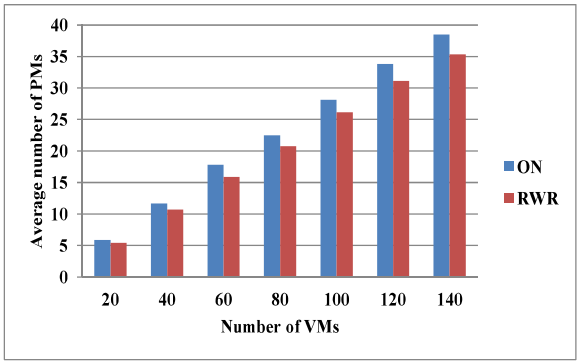


Figure 3.10: Comparison of Number of Physical Machines used

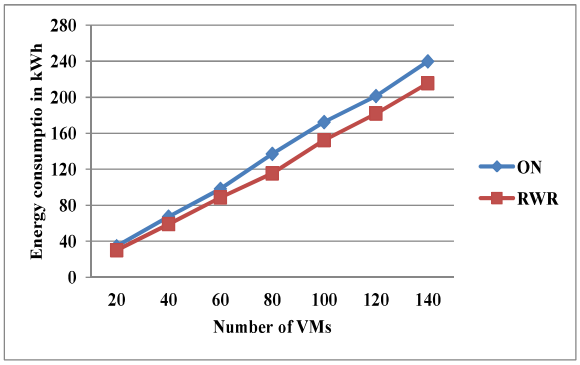


Figure 3.11: Comparison of Energy Consumption

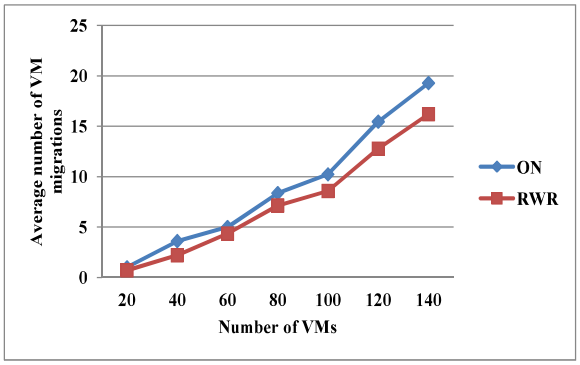


Figure 3.12: Comparison of Number of Migrations

3.6.2 Performance Evaluation in OpenNebula Cloud Environment

The performance of RWR based resource allocation methodology is compared to resource allocation technique used in OpenNebula (ON). OpenNebula based private Cloud environment is configured on forty PMs. Front-end server that executes services of OpenNebula is configured on Lenovo Xeon 1.7 GHz ThinkStation with 8GB

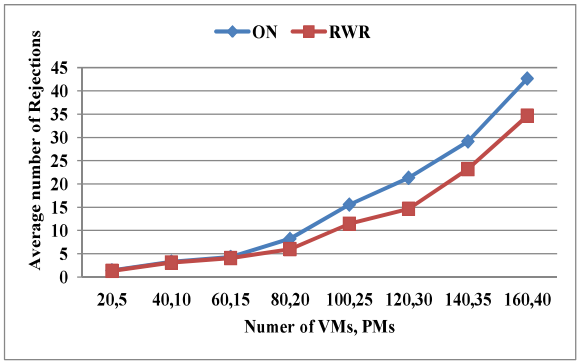


Figure 3.13: Comparison of average number of rejections

RAM and 2TB hard disk. Whereas, OpenNebula software for nodes is installed on thirty nine Quad-core, 8GB RAM, 500GB HDD machines. The physical resources are virtualized leveraging KVM hypervisor. Network File System (NFS) is configured on front-end server to store base images of VMs.

OpenNebula allows the implementation of several resource allocation heuristics through RANK expression [183]. The policy to be used for VM placement can be configured globally for all the VMs. RWR methodology is implemented using RANK expression which ranks PMs using monitoring information gathered from PMs. PMs having insufficient resources to meet the requirements of a VM are filtered out. VMs are allocated to a PM with a higher rank value first. Further, available resource capacity of a PM is checked to ensure that it meets VM requirements. OpenNebula predefined host attributes are used to check available resource capacity [185].

Figure 3.10 shows a comparison of the number of servers (PMs) used by ON and RWR as the number of VMs are varied from 10 to 140. The POLICY parameter in sched. conf file of OpenNebula is set to 0 in order to minimize the number of PMs in use. It is experimentally observed that on an average RWR uses 8.2% lesser number of servers than ON. Lesser number of servers in use indicate improvement in resource utilization. Hence, RWR outperforms ON in terms of resource utilization.

Figure 3.11 shows a comparison of energy consumption of ON and RWR for different number of VMs. The figure shows energy consumption results of 24hrs of run. The average energy consumption of a PM is assumed 230 watts, and only the energy consumption of servers hosting VMs is taken into consideration. It is experimentally established that RWR consumes 11.75% lesser energy than ON. The figure depicts that energy consumption increases with increase in number of VMs, but the rate of

increase of energy consumption in case of RWR is lower than ON.

Figure 3.12 shows a comparison of number of VM migrations in case of ON and RWR. The LIVE_RESCHEDS parameter in sched.conf file is set to 1 to perform live migrations. It is observed that number of VM migrations increases with an increase in the total number of VMs. Approximately 20% lesser VM migrations are observed in RWR than ON. As VM migration degrades quality of service, thus RWR improves QoS provided to the end users.

Figure 3.13 shows a comparison of the number of VMs rejections due to unavailability of sufficient free resources. The number of VMs is varied from 20 to 160. However, number of PMs are varied from 5 to 40. It is observed that ON and RWR perform neck to neck in terms of average number of rejections when the number of VMs are between 20 and 80. As the number of VMs is increased beyond 80, an abrupt increase in the number of rejections occurs due to unavailability of sufficient resources. On an average, approximately 7% lesser rejections are observed in RWR.

3.7 Summary

In this chapter, a Green Cloud framework for efficient and robust management of resources has been presented, and then comparative analysis of the proposed framework with open source as well as real Cloud environments has been conducted. Further, a novel resource allocation approach based on resource wastage reduction methodology has been introduced which judiciously allocates physical machines' resources to improve their utilization and hence the energy efficiency. The performance evaluation of the proposed resource allocation methodology has been carried out on simulated as well as on Open Nebula based private Cloud environment. The next chapter discusses the energy aware resource allocation for Green Cloud.

Chapter 4

Energy Aware Resource Allocation for Clouds[2]

The previous chapter discussed in detail the framework for green cloud for efficient and robust management of resources. This chapter enhances the usefulness of the framework with energy efficient resource allocation approach that not only minimizes total cost of execution, total execution time and total energy consumption but also satisfies QoS requirements of the end users. The proposed approach is based on Ant Colony Optimization (ACO) metaheuristic that is employed at two levels for energy efficient and QoS aware resource allocation. The first level ACO allocates tasks to virtual machines according to the length of the tasks and their associated QoS requirements. The second level ACO decides the placement of virtual machine over physical servers considering the geographical distance and available resource capacity of the physical machines.
This chapter starts with a description of the proposed energy aware resource allocation methodology followed by the derivation of mathematical expressions for total energy consumption and execution cost. The chapter also discusses energy consumption reduction through Dynamic Voltage Frequency Scaling (DVFS). Further, it presents an ACO metaheuristic based energy efficient resource allocation. The chapter concludes with performance evaluation of the proposed energy efficient resource allocation approach.

[2]Ashok Kumar, Rajesh Kumar, Anju Sharma, *"Energy Aware Resource Allocation for Clouds Using Two Level Ant Colony Optimization"*, Computing and Informatics, ISSN 1335-9150, 2016 [in press]

4.1 Energy Aware Resource Allocation Methodology

The proposed Energy Aware Resource Allocation (EARA) methodology allocates resources to jobs using ant colony optimization. The primary goal is efficient utilization of resources in order to reduce energy consumption while respecting Quality of Service (QoS) requirements of the end users' jobs. The jobs are associated with resource and QoS requirements. Each QoS parameter of a job is assigned a weight value. EARA allocates resources for jobs in accordance with their resource demands and weight values of QoS parameters.

The key characteristics of the proposed EARA approach are:

- It deliberately assigns resources to jobs in order to improve the utilization of resources, thereby increases the energy efficiency of the cloud infrastructure.
- Idle Physical Machines (PMs) are switched to sleep mode to conserve energy.
- Monitoring of resource utilization viz. processor, memory, and network bandwidth of a PM is exercised for energy efficient resource allocation and management.
- It supports dynamic performance scaling of the servers to conserve energy.
- Server consolidation is used to minimize the number of active servers.

A brief description of the various components of EARA model, shown in Figure 4.1, is:

- *Cloud Portal*: It provides an interface to the cloud users to input their jobs and desired QoS.
- *Workload Analyzer (WA)*: It analyses QoS requirements of the jobs and classifies them into different classes using k-means clustering algorithm proposed by Kumari *et al.* [186].

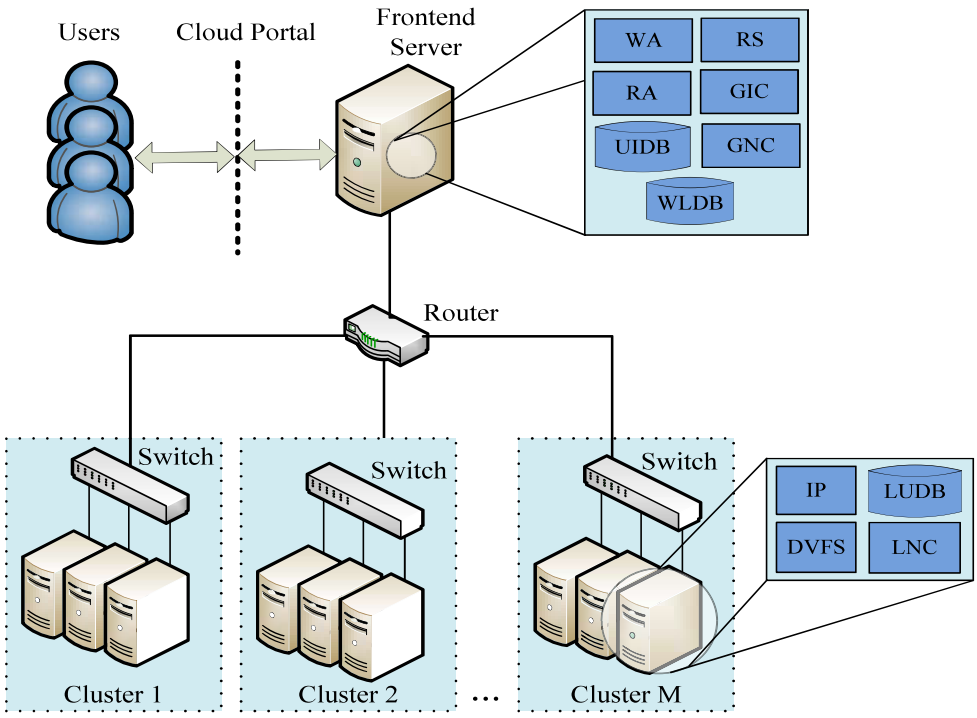


Figure 4.1: Energy aware resource allocation system architecture

- *Resource Scheduler (RS)*: It generates a schedule of jobs to be executed.
- *Resource Allocation (RA)*: It is the core module of the proposed approach, where the resources are allocated to users' jobs as per QoS requirements. It applies ant colony optimization at two levels to allocate: (i) jobs to VMs, and (ii) VMs to PMs. The resources are allocated in accordance with resource demands and weight values of QoS parameters associated with the jobs.
- *Global Information Collector (GIC)*: It receives the resource utilization data from Information Probes (IP) of every PM and stores it in Utilization Information Database (UIDB).
- *Utilization Information Database (UIDB)*: Resource utilization data of each PM are kept in UIDB for future resource allocation and VM migration decisions.
- *Global Node Controller (GNC)*: It initiates live migration of VMs running on a PM when resource utilization of PM violates Lower Green Threshold (LGT) or Upper Green Threshold (UGT) limit.
- *Workload Database (WLDB)*: It stores the information associated with each job.

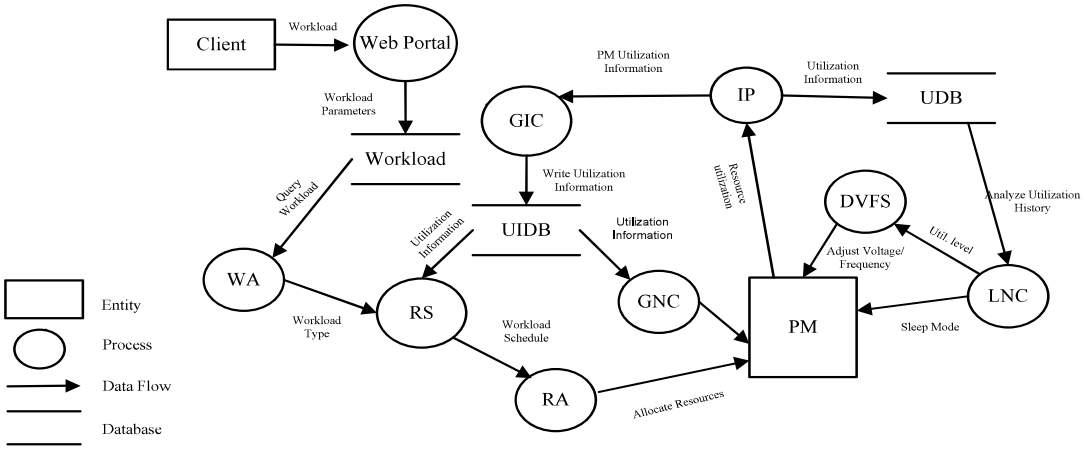


Figure 4.2: Data flow representation for energy aware resource allocation methodology

- *Information Probes (IP)*: Information probes are the specialized programs that executes in host machine to gather typical performance and configuration parameters for the host and VMs such as CPU, network consumption and hostname. The utilization of resources is observed and then recorded in Local Utilization Database (LUDB).

- *Local Utilization Database (LUDB)*: It keeps record of utilization of resources of PMs.

- *Dynamic Voltage Frequency Scaling (DVFS)*: It adjusts the voltage and frequency of the PM in order to reduce power consumption and heat dissipation. The voltage/frequency of PM is adjusted in accordance to resource demands of the VMs/jobs running over it.

- *Local Node Controller (LNC)*: This module uses the utilization history of the node and switch the PM to sleep mode if it is found idle for a specific period of time.

The information flow among various components is depicted in Figure 4.2 through data flow diagram using Yourdon/DeMarco notation [187]. The directional lines (arrows) show information exchange between components. They do not give any information about the timing and sequence of execution of processes. The client inputs jobs from cloud portal. The front-end server stores the job's requirements in the workload

database for analyzing, scheduling, and energy efficient resource allocation. Each job has different resource and QoS requirements. For example, a batch job may require storage and computing resources (i.e. memory, CPU), whereas for online job, network bandwidth may be more critical. WA analyses the jobs and classifies them into different categories based on weight values of their QoS parameters. Jobs are then mapped to VMs. VMs are allocated resources and then scheduled on PMs. Once a VM is deployed on a PM, its resource utilization is monitored by IP after a customizable fixed interval and observed values are stored in LUDB. When utilization of a PM is observed below LGT for two consecutive monitoring intervals, either of the two methods is applied to reduce energy consumption of the PM. First, offloading the lightly loaded PM by migrating all the VM running on it to some other PM and then switching it to sleep mode. Sometimes, it is economically unfeasible to migrate the VMs running on a PM to any other PM, either due to high migration cost, or deadline violation due to migration, or unavailability of PM that can host the VMs due to insufficient available free resources. When VM migration is not economically feasible, the second method for reducing energy consumption, dynamic performance scaling of the PM, is applied. In this technique, voltage/frequency of the PM is intentionally scaled down to reduce its energy consumption. DVFS helps to cut power cost, but at the price of slower job execution. It is applied when the user deadlines can be achieved at slower execution speed.

For implementation and subsequent evaluation of EARA, mathematical equations for energy aware resource allocation, DVFS, and fitness function are formulated as discussed in the forthcoming subsections.

4.1.1 Mathematical Modeling of Energy Aware Resource Allocation

EARA supports energy efficient usage of resources, and is considered from both provider's and client's point of view. It minimizes: (i) total energy consumption for the benefit of service provider, and (ii) total execution cost and total time of execution for the satisfaction of end users.

The following assumptions are taken into considerations while formulating mathematical representation of EARA.

(i). Physical nodes can be unilaterally switched ON/OFF, or put to sleep mode as and when required.

(ii). Every PM supports Advanced Configuration and Power Interface (ACPI). The operating system can adjust voltage and frequency to any level supported by the hardware.

(iii). PMs and VMs are characterized by processing speed, memory, and network bandwidth only.

(iv). Transition from one voltage/frequency level to another is instant.

(v). Energy consumed by a PM during sleep mode is negligible.

(vi). All the cores of a PM can be operated at same voltage/frequency level at a time.

Total power consumption of a PM [71] at any instant is given by equation (4.1).

$$\begin{aligned} P_{total} &= P_{dynamic} + P_{static} \\ &= \underbrace{ACV^2f}_{\text{DPC}} + \underbrace{V * I_{leak}}_{\text{SPC}} \ , \end{aligned} \tag{4.1}$$

where A is switching activity, C is capacitance, V is voltage, I_{leak} is the leakage current, and f is the clock frequency applied to the cores of PM. P_{total}, P_{static}, and $P_{dynamic}$ are total power consumption, Static Power Consumption (SPC), and Dynamic Power Consumption (DPC) of a PM, respectively. SPC is due to leakage current that is present in any active circuit, and it is independent of clock frequency and usage scenario. It can be reduced by switching PM to sleep mode [71]. Whereas, DPC is due to circuit activity and it depends on usage scenario or resource utilization. EARA reduces SPC by switching PM to sleep mode as and when required, whereas DPC is optimized by efficient utilization of resources as explained below.

Suppose s is processing speed, and p is DPC of a PM. Then, $s \propto f$, and $f \propto V$, which implies $p \propto f^3$ and $p \propto s^3$ [188]. Suppose operational requirements of N_t tasks

(jobs) can be fulfilled by a N_v number of VMs, and each VM has the resources to fulfill needs of n_g tasks. R_g is execution requirement, and E_{ijg} is total energy consumption of the tasks deployed on g^{th} VM that is instantiated on j^{th} PM of cluster i. If r_{ij}^q is the execution requirement of task q on this PM, then, the execution time t_{ij}^q of the task q on this PM with power p_{ij}^q and processing speed s_{ij}^q can be calculated from equation (4.2).

$$t_{ij}^q = \frac{r_{ij}^q}{s_{ij}^q} = \frac{r_{ij}^q}{(p_{ij}^q)^{\frac{1}{3}}} \ , \tag{4.2}$$

and the energy consumption of this PM to execute task is given by e_{ij}^q as shown in equation (4.3).

$$e_{ij}^q = p_{ij}^q . t_{ij}^q = r_{ij}^q (p_{ij}^q)^{\frac{2}{3}} = r_{ij}^q (s_{ij}^q)^2. \tag{4.3}$$

With time constraint T_g, the objective is to:

(i). Minimize energy consumption (E_{ijg}) of g^{th} VM

(ii). Restrict the execution time of all tasks $T_{ijg} = t_{ij}^1 + t_{ij}^2 + ... + t_{ij}^{n_g}$ to deadline time T_g.

Energy consumption (E_{ijg}) and execution time (T_{ijg}) are calculated as shown in equation (4.4)

$$E_{ijg}(p_{ij}^1, p_{ij}^2, ..., p_{ij}^n) = r_{ij}^1 {p_{ij}^1}^{\frac{2}{3}} + r_{ij}^2 {p_{ij}^2}^{\frac{2}{3}} + ... + r_{ij}^{n_g} {p_{ij}^{n_g}}^{\frac{2}{3}}, \tag{4.4}$$

and equation (4.5)

$$T_{ijg}(p_{ij}^1, p_{ij}^2, ..., p_{ij}^n) = \frac{r_{ij}^1}{{p_{ij}^1}^{\frac{1}{3}}} + \frac{r_{ij}^2}{{p_{ij}^2}^{\frac{1}{3}}} + ... + \frac{r_{ij}^{n_g}}{{p_{ij}^{n_g}}^{\frac{1}{3}}} \leq T_g. \tag{4.5}$$

As both energy consumption and execution time are function of $p_{ij}^1, p_{ij}^2, ..., p_{ij}^{n_g}$. So, solving equation (4.4) and (4.5) using Lagrange multiplier system [189], we get execution time as represented in equation (4.6)

$$T_{ijg} = \frac{R_g^{\frac{3}{2}}}{\sqrt{E_{ijg}}}, \tag{4.6}$$

and energy consumption, as shown in equation (4.7)

$$E_{ijg} = P_{ijg}T_{ijg} = \left(\frac{R_g}{T_{ijg}}\right)^3 T_{ijg} = \frac{R_g^3}{(T_{ijg})^2}. \tag{4.7}$$

where $R_g = r_{ij}^1 + r_{ij}^2 + ... + r_{ij}^{n_g}$, is a total execution requirement of all the tasks deployed on g^{th} VM.

Total Energy Consumption (TEC) by all the VMs can be calculated as shown in equation (4.8).

$$TEC \quad = \sum_{g=1}^{N_v} E_{ijg}, \tag{4.8}$$

Total Execution Time (TET) of N_t tasks running on N_v VMs can be calculated as shown in equation (4.9).

$$TET = T_{ij1} + T_{ij2} + ... + T_{ijN_v}. \tag{4.9}$$

In order to minimize TET and TEC, every VM have to be carefully mapped to the suitable PM of a cluster.

Suppose a set $CL = \{cl_i | 1 \leq i \leq N_c\}$ of clusters, each cluster i has a pool of physical machines $PM_i = \{h_{ij} | 1 \leq j \leq H_i\}$. Here, j^{th} PM of cluster i has computing power represented by $h_{ij} = \{hs_{ij}, hm_{ij}, hn_{ij}\}$, where hs_{ij}, hm_{ij}, and hn_{ij} is CPU processing speed, memory, and network bandwidth, respectively. VM computing requirements are represented by $v_g = \{vs_g, vm_g, vn_g\}$, where vs_g, vm_g, and vn_g are processing speed, memory, and network bandwidth of g^{th} VM.

Suppose C_{ijg} is total cost (processing cost+Memory cost+Bandwidth cost) of running g^{th} VM on j^{th} PM of cluster i, and x_{ijg} equal to 1 if g^{th} VM is assigned to j^{th} PM of cluster i and 0 otherwise.

Execution cost (EC_{ij}) of all the VMs running on j^{th} PM of cluster i can be calculated from equation (4.10)

$$EC_{ij} = C_{ijg} * x_{ijg} \ , \forall g | x_{ijg=1}, \tag{4.10}$$

and Total Cost of Execution (COE) of all the VMs can be evaluated using equation (4.11)

$$COE \quad = \sum_{i=1}^{N_c} \sum_{j=1}^{H_i} EC_{ij} \tag{4.11}$$

A VM should be assigned to PM in such a way that all the three conditions, represented by equation (4.12), (4.13), and (4.14) are satisfied.

$$\sum_{g=1}^{N_v} vs_{ijg} * x_{ijg} \leq hs_{ij} * CPU_{UGT}, \qquad \forall i,j | x_{ijg} = 1 \ , \tag{4.12}$$

$$\sum_{g=1}^{N_v} vm_{ijg} * x_{ijg} \leq hm_{ij} * MEM_{UGT}, \qquad \forall i,j | x_{ijg} = 1 \ , \tag{4.13}$$

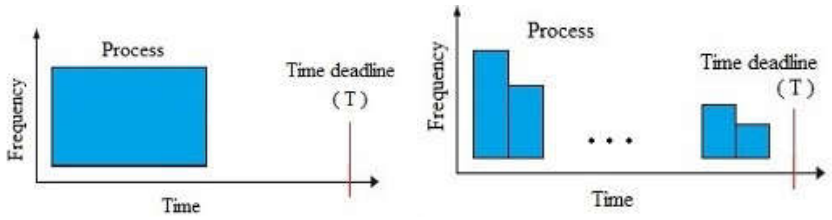
$$\sum_{g=1}^{N_v} vn_{ijg} * x_{ijg} \leq hn_{ij} * BW_{UGT}, \qquad \forall i,j | x_{ijg} = 1. \tag{4.14}$$

These three conditions put a cap on the maximum utilization of PM resources. Utilization of resources determines power consumption of a PM. In fact, the power consumed by a PM increases linearly with its utilization [94]. EARA uses relationship between power consumption of a PM and its utilization (equation (4.15)) as a basis for energy consumption calculation.

$$P = P_{idle} + (P_{max} - P_{idle})U, \tag{4.15}$$

where P_{idle} is the power consumption when PM is idle, P_{max} is the power consumption at 100% utilization, and P is the power consumption at utilization $U \in (0, 1)$ of the PM. An idle PM consumes 70% of the power consumed at full load [94]. Whereas, high utilization of PM resources causes performance degradation because tasks running on it do not get sufficient resources [94]. So, PM should not the operated at too low or very high utilization in order to improve its energy efficiency.

EARA assigns PM resources to VMs using ACO. Resource allocation using ACO is discussed in section 4.2. Besides optimal allocation and management of PM resources, DVFS as in Wu *et al.* [121] is applied to further reduce the energy consumption of a PM. In DVFS technique, energy is conserved by intentionally scaling down the performance of a PM as discussed in the next subsection.



(a) Execution at maximum frequency (b) Execution at different frequencies

Figure 4.3: Dynamic voltage frequency scaling

4.1.2 Dynamic Voltage Frequency Scaling

Modern processors support ACPI, and thus can be operated at different levels of voltage/frequency. This feature of modern processors has been extensively used in [121, 188–192] to reduce energy consumption as well as the heat dissipated. DVFS is used to execute a process/task using different combination of frequencies/voltages to conserve energy. Figure 4.3 shows the effect of executing a task using different combination of supported frequencies. Figure 4.3(a) depicts the case of executing a process/task at maximum frequency. The process may finish well in advance of its deadline time T. The completion of a task sometimes lowers the utilization of a PM below LGT. In such cases, energy consumption can be reduced by either server consolidation or dynamic performance scaling of PM. Sometimes consolidation is not economically feasible due to constraints such as communication cost of migration, QoS violations due to interruption in service while consolidating, or unavailability of PM with sufficient free resources where the VM can be migrated. In such cases energy consumed by PMs can be saved by executing the process using different combinations of voltage/frequency [121, 188–192] as shown in Figure 4.3(b). In EARA, when the utilization of a PM drops below LGT and remains less than LGT for two consecutive monitoring periods, voltage/frequency of PM is adjusted to lower supported level to conserve energy. In case of low utilization, LNC sends current utilization value 'U' to DVFS module as shown in Figure 4.4. DVFS module calculates voltage/frequency level 'VF' that can complete the workload before user deadline. Voltage/frequency of PM is then adjusted to 'VF' to conserve energy. Energy consumption is reduced at the cost of slower process execution. Elongated process execution time does not affect user satisfaction

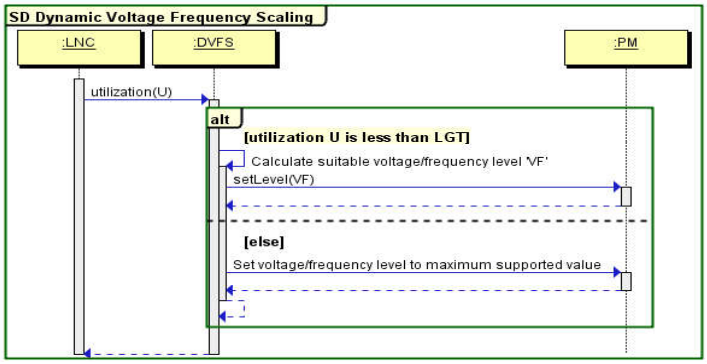


Figure 4.4: Sequence diagram for dynamic voltage frequency scaling

because frequencies are selected in such a manner to complete the process before desired time deadline. The different frequencies are calculated to complete the task by deadline time T. The PMs in EARA are assumed to have ACPI support and can be operated at any of the discrete h frequencies $f_1 < f_2 < ... < f_{h-1} < f_h$, supported by it. The key idea behind applying DVFS is to execute tasks using linear combination of supported frequencies. The minimization of power consumption is formulated as shown in equation (4.16).

$$\begin{cases} Min\ E_{ij} = \sum_{l=1}^{h} t_l A C V^2 f_l\ , \\ s.t. \\ \quad \sum_{l=1}^{h} t_l \leq T \\ \quad t_l \geq 0, \qquad\qquad\qquad \text{for } l = 1, 2, ..., h. \end{cases} \tag{4.16}$$

Where h is the number of supported frequencies, t_l is the time period for which the task is executed at frequency f_l.

4.1.3 Fitness Function

The main goal is to minimize total energy consumption, total execution time, and cost of execution by efficient utilization of resources. The weighted sum method is used to scalarize multiple objectives into a single objective. The weighted fitness function is

calculated as shown in equation (4.17).

$$F(N_t) = \xi(TEC_{min}) + \zeta(TET_{min}) + \gamma(COE) \ , \tag{4.17}$$

where,

$$TEC_{min} = \sum_{g=1}^{N_v} min(E_{ijg}), \quad \forall i, j \ , \tag{4.18}$$

$$TET_{min} = \sum_{g=1}^{N_v} min(T_{ijg}), \quad \forall i, j \ , \tag{4.19}$$

$0 \leq \xi, \zeta, \gamma < 1$, and $\xi + \zeta + \gamma = 1$. The value of weights ξ, ζ, and γ depends on the importance of each objective in the context of resource allocation problem.
Mathematical model formulated for EARA is implemented using ACO metaheuristic as discussed in the next section.

4.2 Ant Colony Optimization based Energy Aware Resource Allocation

Ant colony optimization is a metaheuristic optimization technique proposed by Marco Dorigo in 1992 [193]. Ants secrete a chemical substance called pheromone while foraging. Pheromone gets deposited on the paths followed by the ants. The amount of pheromone deposited on the path depends on the number of ants that followed the path. So a path that is used by more number of ants will have higher quantity of pheromone deposit. Initially, the ants choose the random path while searching for food. But with time the difference of quantity of pheromone deposited on the paths guide them to choose the path marked with strong pheromone concentration. The larger amount of pheromone on a path attracts more ants to choose that path again and finally all the ants converge to a single path. Further, pheromone evaporates with time, thus reducing the attractive strength of the path, and causing ants to explore more paths to the food source. Pheromone evaporation has the advantage of avoiding the convergence to locally optimal solution.
The reasons behind choosing ACO for resource allocation are:

(i). It can solve certain NP-hard problems in polynomial time.

(ii). It maintains a balance between acquired knowledge and exploring new solutions exploiting pheromone evaporation.

(iii). It gives near to optimal solution.

(iv). It performs distributed computation to avoid premature convergence.

Ant colony optimization has been applied to solve a wide range of combinatorial optimization problems [100, 111, 112, 136, 193, 194]. To solve a combinatorial optimization problem using ant colony optimization, an instance of the problem has to be mapped to a graph $G = (N, L)$, called construction graph. Node sets N of the graph represents the components of the problem instance and edge set L fully connects the components. Each edge (u, v) of the graph is associated with pheromone trail τ_{uv} and heuristic information η_{uv}. Heuristic information can be cost, distance, etc. Each ant uses pheromone trail and heuristic information, probabilistically, to construct its own solution of the problem instance. Once the solution is constructed, the pheromone trail associated with every edge is updated to reflect evaporation, in order to enable ants to forget previously taken bad decision. The pheromone trail on each edge that belong to the best solution is then updated.

For energy efficient resource allocation ant colony optimization is applied at two levels: (i) First level ACO allocates jobs to VMs, and (ii) Second level ACO allocates of PM resources to VMs. Detailed description of graph construction, pheromone and heuristic information, solution construction, pheromone evaporation, and pheromone trail updation for allocation of jobs to VMs, and VMs to PMs as follows:

4.2.1 Allocation of Virtual Machine Resources to Jobs

Each job has some resource demands and QoS requirements. Further, a QoS parameter is assigned with some weight value that indicates its priority over the others. The QoS parameters can be assigned weights in three ways: absolute weight, relative weight, and arbitrary weight [195]. In EARA, relative weight is used for QoS attributes.

The first level ACO metaheuristic for allocation of VM resources to jobs is as explained below:

Construction Graph: The problem of allocation of VM resource to jobs is mapped to construction graph $G_1 = (N_1, L_1)$. The node set N_1, consists of all VMs and jobs. The set L_1 of edges fully connects the nodes of the graph G_1. Each edge (a, g) of the graph G_1 is assigned pheromone value given by equation (4.20).

$$\tau_{ag} = \frac{1}{\ell_a} \ , \tag{4.20}$$

where a is the unique identification number of a job, g is the unique identification number of a VM, ℓ_a is the length of the job a. In general, length of job (cloudlet) is in millions of instructions. As the inverse of job length is used as pheromone value so shorter jobs will be given preference over the longer ones.

Heuristic information assigned to edge (a, g) is given by equation (4.21).

$$\eta_{ag} = \prod_{x=1}^{X} \left(W_{ax} \right)^{\alpha_x} \ , \tag{4.21}$$

where X is number of QoS parameters associated with job a, W_{ax} is the weight value of QoS parameter x, and α_x is a control parameter for QoS attribute x of job a.

Solution Construction: Each ant is initially provided with the list of jobs to be mapped on VMs. For each job a that is mapped to g^{th} VM, variable y_{ag} is set to 1. The variable y_{ag} is used to keep a record of jobs that have already been assigned. The probability that an ant k maps job a on g^{th} VM is given by

$$\wp_{ag}^{k} = \begin{cases} \frac{(\tau_{ag})^{\alpha 1}(\eta_{ag})^{\beta 1}}{\sum_{t \in \mathcal{N}_g^k}(\tau_{tg})^{\alpha 1}(\eta_{tg})^{\beta 1}}, & \text{if } t \in \mathcal{N}_g^k \ , \\ 0 \ , & \text{otherwise,} \end{cases} \tag{4.22}$$

where $\mathcal{N}_g^k$, consisting of the jobs to be mapped, is called feasible neighborhood of g^{th} VM. A job a which have a maximum value of $\wp_{ag}^k$ is mapped to g^{th} VM. The probability of a job selection for mapping on a particular VM depends on the value of pheromone trail and heuristic information of the associated edge. $\alpha 1$ is the parameter to control the contribution of pheromone trail in the VM selection, and $\beta 1$ is a parameter to control the overall contribution of QoS parameters. Both $\alpha 1$ and $\beta 1$ can

have any value between 0 and 1, and their sum should be equal to one.

Pheromone Trail Evaporation: The pheromone deposit on the arcs evaporates with time by a constant factor $0 \leq \rho_1 \leq 1$, called evaporation rate. Evaporation avoids unlimited accumulation of pheromone trails on the edges and enables ants to forget previously taken allocation decisions. Pheromone evaporation on the edges of graph G_1 is defined by equation (4.23).

$$\tau_{ag} = (1 - \rho_1)\tau_{ag}, \ \forall (a, g) \in L_1. \tag{4.23}$$

Pheromone Trail Updation: In order to reflect usage of an arc during solution construction, the pheromone trail on it is updated by an amount equal to the inverse of the number of VMs used by an ant for solution construction. Updation makes pheromone concentration on some of the arcs stronger than the others. Strong pheromone concentration on an edge increases the probability of selection of the associated job. Pheromone updation by ant k is realized by equation (4.24).

$$\tau_{ag} = \tau_{ag} + \sum_{k=1}^{N_a^1} \Delta\tau_{ag}^k, \ \forall (a, g) \in S1^k, \tag{4.24}$$

where N_a^1 is the number of ants, $\Delta\tau_{ag}^k$ is the amount of additional pheromone to be deposited on edge (a, g) which is traversed by ant k while constructing the allocation solution, and $S1^k$ is jobs to VMs mapping solution constructed by ant k. The amount of pheromone deposited on arc (a, g) by ant k is defined as follows:

$$\Delta\tau_{ag}^k = \begin{cases} \frac{1}{D1^k} \ , & (a, g) \in S1^k \ , \\ 0, & \text{otherwise}, \end{cases} \tag{4.25}$$

where $D1^k$ is number of VMs used for mapping of all jobs and calculated as length of solution $S1^k$.

4.2.2 Allocation of Physical Machines Resources to Virtual Machine

Construction Graph: The resource allocation problem is mapped to construction graph $G_2 = (N_2, L_2)$. The node set N_2, consists of all VMs and PMs. L_2 is set of edges that fully connects the nodes. Each edge (g, j) of the graph G_2 is associated with pheromone trail τ_{gj} and heuristic information η_{gj}, where g is the identification number of a VM and j is the identification number of a PM.

The pheromone trail associated with an edge(g, j) is given by equation (4.26)

$$\tau_{gj} = \frac{1}{\frac{vm_g}{FM_j}} \ , \tag{4.26}$$

where vm_g is memory requirements of g^{th} VM, and FM_j is available memory space of j^{th} PM which can be calculated for given g and j using equation (4.27)

$$FM_j = hm_{ij} - \sum_{g} vm_{ijg} * x_{ijg}, \forall i, j | x_{ijg} = 1. \tag{4.27}$$

Heuristic information assigned to edge (g, j) for j^{th} PM of cluster i is given by equation (4.28)

$$\eta_{gj} = \frac{1}{d_{ij}}, \tag{4.28}$$

where d_{ij} is the distance between the front-end server and j^{th} PM of cluster i.

Solution Construction: Each ant is initially provided with the list of VMs to be deployed. x_{ijg} is set to 1 for each VM g that is assigned to j^{th} PM of cluster i. Variable x_{ijg} is used to keep record of VM that have already been assigned. The probability that an ant k deploys g^{th} VM on j^{th} PM of cluster i is given by

$$\wp_{ijg}^{k} = \begin{cases} \frac{(\tau_{gj})^{\alpha 2}(\eta_{gj})^{\beta 2}}{\sum_{t \in \mathcal{N}_j^k}(\tau_{tj})^{\alpha 2}(\eta_{tj})^{\beta 2}}, & \text{if } t \in \mathcal{N}_j^k \ , \\ 0 \ , & \text{otherwise,} \end{cases} \tag{4.29}$$

where $\mathcal{N}_j^k$ is the feasible neighborhood of j^{th} PM, comprising all those VMs which can still be deployed on it. The probability of choosing j^{th} PM for g^{th} VM increases with

an increase in the value of associated pheromone trail τ_{gj} and heuristic information η_{gj}. The parameters $\alpha 2$ and $\beta 2$ control the influence of pheromone trail and heuristic information on PM selection, and can have any value between 0 and 1.

Pheromone Trail Evaporation: The pheromone deposited on the arcs of graph G_2 evaporates with time by a constant factor, $0 \leq \rho_2 \leq 1$, called evaporation rate. Evaporation avoids unlimited accumulation of pheromone trails on the edges and enables ant to forget allocation decisions previously taken. Pheromone evaporation on all the edges of graph G_2 is realized by equation (4.30).

$$\tau_{gj} = (1 - \rho_2)\tau_{gj}, \ \forall (g, j) \in L_2. \tag{4.30}$$

Pheromone Trail Updation: In order to reflect usage of an arc during solution construction, the pheromone trail on it is updated by an amount equal to the inverse of the length of the solution path. Updation makes pheromone concentration on some of the arcs stronger than the others. Strong pheromone concentration increases the probability of arc selection, which is exploited to optimize the utilization of PM. Pheromone updation by ant k is realized by equation (4.31):

$$\tau_{gj}^k = \tau_{gj}^k + \sum_{k=1}^{N_a^2} \Delta\tau_{gj}^k, \ \forall (g, j) \in S2^k, \tag{4.31}$$

where N_a^2 is the number of ants, $\Delta\tau_{gj}^k$ is the amount of additional pheromone to be deposited on arc (g, j) which is traversed by ant k while constructing the solution, and $S2^k$ is solution constructed by ant k, consisting of VMs to PMs mapping.

The amount of pheromone deposited on arc (g, j) by ant k is defined as equation (4.32):

$$\Delta\tau_{gj}^k = \begin{cases} \frac{1}{D2^k} \ , & (g, j) \in S2^k \ , \\ 0, & \text{otherwise}, \end{cases} \tag{4.32}$$

where $D2^k$ is computed as the sum of the lengths of the arcs belonging to solution $S2^k$.

Algorithm 4.1 EARA

```
1: PROCEDURE EARA(VM)
2: initialization
3: while not Termination Condition do
4:     Allocation Solution Construction
5:     Pheromone Evaporation
6:     Pheromone Trail Updation
7: end while
8: return VM to PM map
```

Algorithm 4.2 *Initialization*

```
1: PROCEDURE Initialization
2: Create construction graph G2(N2, L2) of resource allocation problem
3: Set VM = ID of nodes representing VMs in the construction graph G2(N2, L2)
4: Set PM = ID of nodes representing PMs in the construction graph G2(N2, L2)
5: for all g in VM do
6:     for all j in PM do
7:         Set τ_gj = 1 / (vm_g / FM_j)
8:         i=getClusterID(j)
9:         Set η_gj = 1 / d_ij
10:    end for
11: end for
```

4.2.3 Algorithms for Energy Aware Resource Allocation

The pseudo code for EARA using ACO has shown in Algorithm 4.1, comprised of four phases: *initialization, allocation solution construction, pheromone evaporation*, and *pheromone trail updation*. As discussed earlier ACO is applied at two levels. At first level, ACO is applied to allocate VM resources to jobs, and at the second level, it is applied to allocate PM resources to VMs. This chapter discusses only the pseudo code for allocation of PM resources to VMs. The given pseudo code can be easily modified for allocation of VM resources to jobs.

For allocation of PM resources to VMs, list of VMs is passed to the procedure EARA (Algorithm 4.1). As shown in Algorithm 4.2, in the *initialization* phase, construction graph of the problem is created and every edge of the graph is assigned initial pheromone trail and heuristic information as discussed earlier. The function getClusterID(), in line number 8, is used to get the cluster ID of a PM. Algorithm 4.3 outlines *allocation solution construction* phase. It probabilistically maps a VM to the appropriate PM. A PM is selected randomly and VMs from its feasible neighborhood are assigned to it one by one till UGT is observed. The process is repeated for each ant until all the VMs are mapped. When all the ants have finished assigning VMs to

Algorithm 4.3 *Allocation Solution Construction*

PROCEDURE **Resource Allocation(k)**
input: Construction graph $G_2(N_2, L_2)$ of the resource allocation problem
Set $S2^k$ = NULL
while not (Are all VMs Alloted?) **do**
  if Available(activePM) **then**
    Set j= ID of randomly selected PM from the list of active PMs
    i=getClusterID(j)
  else
    Set j= ID of randomly selected PM from list of newly added PMs
    i=getClusterID(j)
  end if
  Set $\mathcal{N}_j^k$ = NULL
  Set VMIDs = IDs of VMs yet not assigned to any PM
  for all g in { VMIDs } **do**
    if j^{th} PM fulfills processing, memory and network bandwidth requirements of g^{th} VM **then**
      Add g^{th} VM to $\mathcal{N}_j^k$
      Evaluate $\wp_{ijg}^k = \frac{(\tau_{gj})^{\alpha 2}(\eta_{gj})^{\beta 2}}{\sum_{t \in \mathcal{N}_j^k}(\tau_{tj})^{\alpha 2}(\eta_{tj})^{\beta 2}}$,
    end if
  end for
  VM_{id}^f= NULL
  P^f=0
  for all g^{th} VM in $\mathcal{N}_j^k$ **do**
    if $\wp_{ijg}^k > P^f$ **then**
      VM_{id}^f=g
      $P^f = \wp_{ijg}^k$
    end if
  end for
  if VM_{id}^f # NULL **then**
    Set $g = VM_{id}^f$
    Set $x_{ijg} = 1$
    Add x_{ijg} to $S2^k$
  end if
end while
return ($S2^k$)

Algorithm 4.4 *Pheromone Evaporation*

PROCEDURE **PheromoneEvaporation()**
Set VM = ID of nodes representing VMs in the construction graph $G_2(N_2, L_2)$
Set PM = ID of nodes representing PMs in the construction graph $G_2(N_2, L_2)$
for all g in VM **do**
  for all j in PM **do**
    $\tau_{gj} = (1 - \rho_2)\tau_{gj}$
  end for
end for

Algorithm 4.5 *Pheromone Trail Updation*

1: PROCEDURE **PheromoneUpdation(k)**
2: **input:** k
3: $D2^k$= Length($S2^k$)
4: $\Delta\tau^k = \frac{1}{D2^k}$
5: **for all** s in $S2^k$ **do**
6: g = getVMID(s)
7: j = getPMID(s)
8: $\tau_{gj} = \tau_{gj} + \Delta\tau^k$
9: **end for**

PMs, pheromone trail evaporation on all the edges is performed by Algorithm 4.4. The pheromone trail on every edge used by an ant for solution construction is then updated as shown in Algorithm 4.5.

4.3 Performance Evaluation and Comparative Analysis

The performance evaluation of EARA is performed on CloudSim. CloudSim is a framework for modeling and simulation of cloud computing infrastructure and services [176]. The CloudSim simulation toolkit is used for implementation, testing, and validation because of the large-scale nature of the real cloud environment. For performance analysis, EARA is compared with existing resource allocation algorithms, i.e. First-Fit-Decreasing (FFD) [196] and Multi-objective Grouping Genetic Algorithm (MGGA) [126]. Five data centers are created with specification as shown in Table 4.1. In each data center, PMs with specifications as shown in Table 4.2 are created. To conduct comparative analysis, four types of VMs as shown in Table 4.3 are used. FFD, MGGA and EARA are tested with jobs having different QoS requirements.

Table 4.1: Specification of Data Centers

Name	*PC*	*MC*	*SC*	*BC*	*Time Zone*
DC1	3	0.05	0.10	0.10	3.0
DC2	3.5	0.07	0.10	0.11	5.0
DC3	4	0.09	0.10	0.07	5.5
DC4	5	0.10	0.10	0.13	8.0
DC5	5.25	0.12	0.10	0.15	10.0

Name, Data center name; PC, Processing cost; MC, Memory cost per MB; SC, Storage cost per MB; BC, Bandwidth cost; Time Zone, Time zone of data center location

Table 4.2: Specification of Physical Machines

PM Type	*CPU*	*Cores*	*RAM*	*Storage*	*BW*
1	1000	4	8	2	10
2	1500	8	16	2	10
3	2000	12	32	2	10
4	3000	20	64	4	10
5	5000	36	64	4	10

CPU, processing speed in mips; Cores, number of processing cores; RAM, random access memory in GB; Storage, Permanent storage capacity in TB; BW, network bandwidth in gbps.

Table 4.3: Specification of Virtual Machines

VM Type	*CPU*	*PEs*	*RAM*	*BW*
1	500	1	512	1
2	1000	2	1024	2
3	2000	4	2048	4
4	4000	8	4096	8

CPU, processing speed in mips; PEs, number of cores; RAM, random access memory in mega bytes; BW, network bandwidth in gbps.

Table 4.4: Simulation Parameters

Number of jobs	200-1200	Varied in every simulation run
Number of datastores	2-5	Stores instances of VMs
Number of ants	10-50	Construct allocation solution
Idle Time	10 min.	Time to switch PM to sleep mode
CPU_{UGT}	0.85	UGT for CPU utilization
MEM_{UGT}	0.85	UGT for memory utilization
BW_{UGT}	0.85	UGT for bandwidth utilization
CPU_{LGT}	0.30	LGT for CPU utilization
MEM_{LGT}	0.30	LGT for memory utilization
BW_{LGT}	0.30	LGT for bandwidth utilization

The aim of executing jobs with different QoS requirements is to test the effectiveness of EARA in terms of energy efficiency, number of physical machines required, and quality of service. Performance of EARA is analyzed by varying the number of jobs in every simulation run. Simulation is repeated 25 times and in each simulation run, parameters are set to a value from the range of values given in Table 4.4.

4.3.1 Comparative Analysis

Figure 4.5 shows a comparison of number of PMs used by FFD, MGGA, and EARA to fulfill the computational requirements of given number of jobs. EARA outperforms

both FFD and MGGA in terms of number of PMs used to deploy given number of jobs. On an average, EARA uses 11.36% and 7.68% lesser number of PMs than FFD and MGGA, respectively.

Figure 4.6 shows a comparison of total energy consumed by FFD, MGGA, and EARA. Total energy consumption of EARA is less than FFD and MGGA because it uses a lesser number of PMs to deploy given number of jobs. It is experimentally established that EARA is 10.56%, and 5.43% more energy efficient than FFD and MGGA, respectively.

Figure 4.7 depicts a comparison of percentage resource utilization of PMs by FFD, MGGA, and EARA. In case of EARA, utilization of 85% of PMs is between 41-80%. In case of FFD and MGGA utilization of more than one third of PMs is above 80% which causes performance degradation of user applications and results in the creation of hot-spots. Moreover, only 2% of the PMs are there in EARA whose utilization is 0-20% as compared to 5% and 7% in FFD and MGGA, respectively. Therefore, EARA is capable of managing the resources efficiently.

Figure 4.8 shows the comparison of the average number of VM migrations. Number of migrations in EARA are less than MGGA but more than FFD. In EARA, when the utilization of a PM falls below LGT value migration of VMs running over it is performed. Migration is performed for PMs consolidation so that some of the PMs can be switched to sleep mode to conserve energy. Approximately two migrations for 1200 jobs are observed, such a small number of migrations does not impact the performance of the system. Moreover, EARA compensates the energy loss due to migrations by switching idle PMs to sleep mode.

Figure 4.9 shows the comparison of number of hot-spots created by FFD, MGGA, and EARA as the number of jobs vary from 200 to 1200. We define a PM as a hot-spot if its resource utilization is 100%. Maximum number of hot-spots are created by FFD methodology because it tries to utilize PM to its full capacity. However, EARA does not create any hot-spot because it keeps resource utilization of PMs between LGT and UGT. Hot-spots adversely affect the performance and reliability of PM. Moreover, the creation of hot-spots demands better cooling arrangements and also increases chances of hardware failure. Hence, EARA is more reliable and energy efficient.

Figure 4.10 shows a comparison of percentage workload of data centers, when $\alpha 2 = 0$.

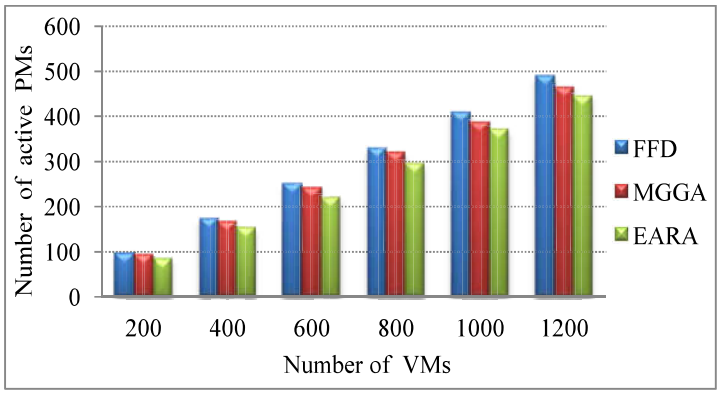


Figure 4.5: Comparison of number of PMs required by FFD, MGGA, and EARA

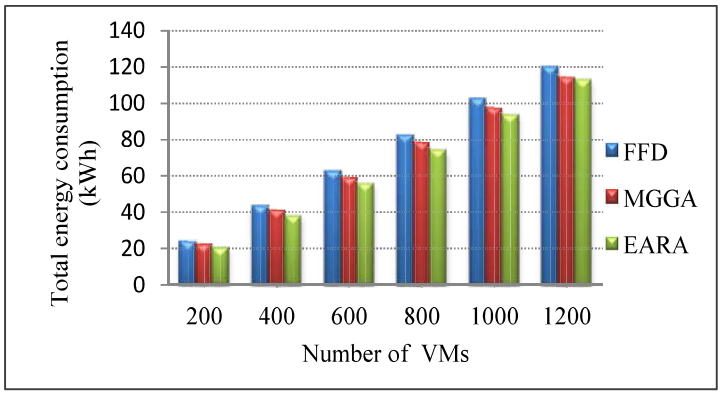


Figure 4.6: Comparison of total energy consumption by FFD, MGGA, and EARA

All the data centers have exactly same computing infrastructure, but are at unequal distances from front-end server. The distance between a datacenter and front-end server is calculated from the time zone of the datacenter. When $\alpha 2 = 0$, EARA gives weightage to distance while mapping VMs to PMs. As distance of DC1 is least, so EARA distributes the VMs to PMs of DC1 first. Once DC1 resources are used upto UGT of their capacity, EARA starts assigning jobs to PMs of the datacenter whose distance is next to a distance of DC1 and so on. EARA saves energy by deploying jobs/VMs over PMs which are nearer to the frontend server because more energy is consumed to transmit jobs/VMs over longer distances. Moreover, deploying VM over nearer datacenter also improves response time.

Figure 4.11 shows a comparison of percentage workload of data centers, when $\beta 2 = 0$. All the data centers are having five types of PMs with specification as per Table 4.2, and the data centers have equal number of specific types of PMs. In case of EARA, variance of the percentage workload of data centers is less than that of FFD

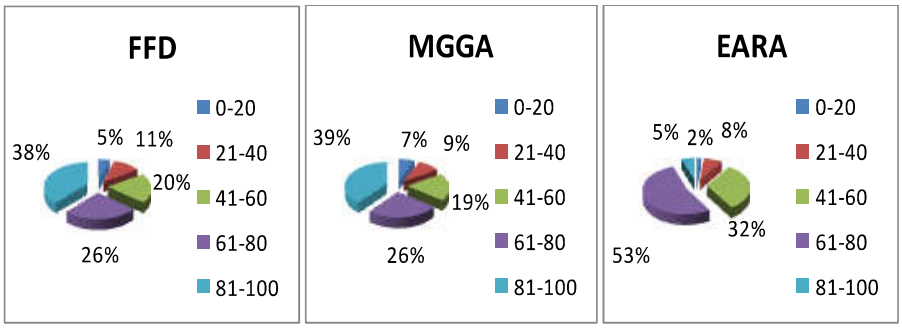


Figure 4.7: Comparison of PMs utilization in FFD, MGGA, and EARA

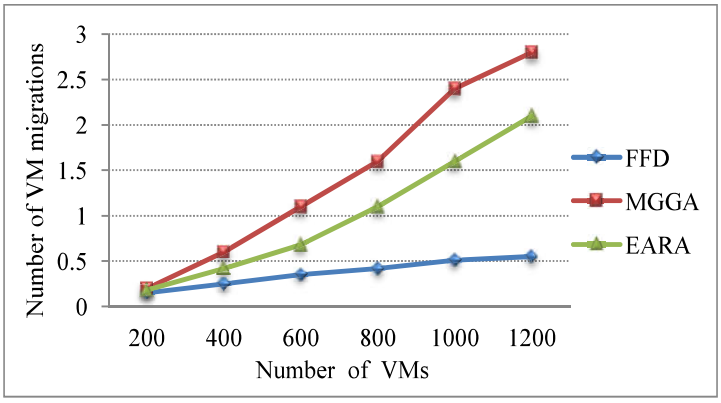


Figure 4.8: Comparison of number of VM migrations in FFD, MGGA, and EARA

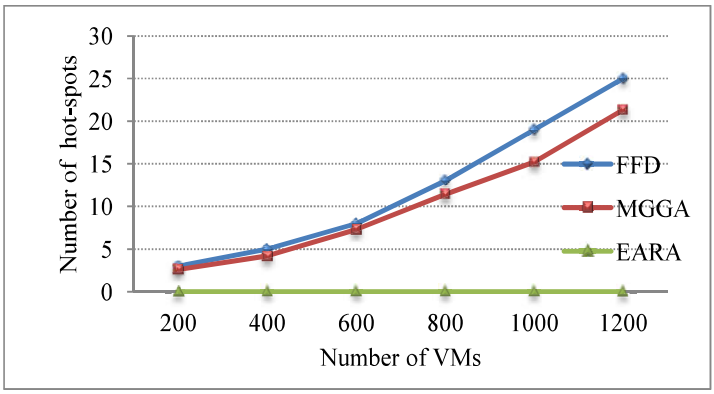


Figure 4.9: Comparison of number of hotspots in FFD, MGGA, and EARA

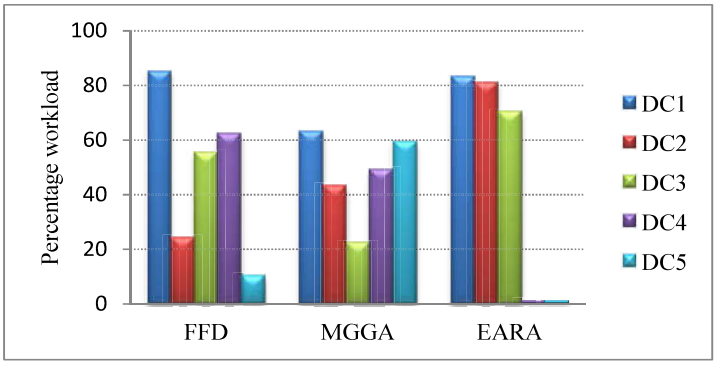


Figure 4.10: Comparison of percentage workload of data centers in FFD, MGGA, and EARA, when $\alpha 2 = 0$

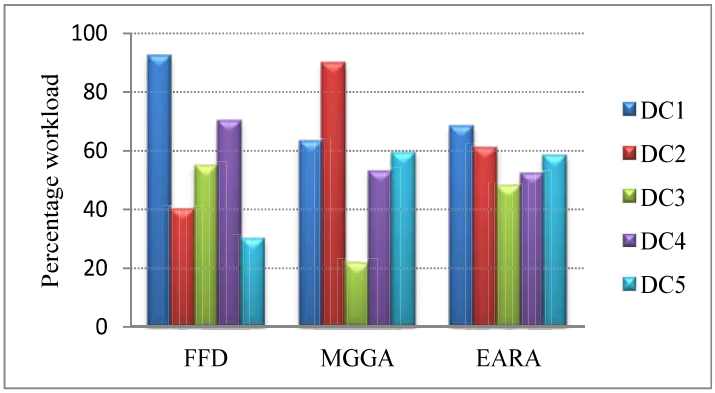


Figure 4.11: Comparison of percentage workload of data centers in FFD, MGGA, and EARA, when $\beta 2 = 0$

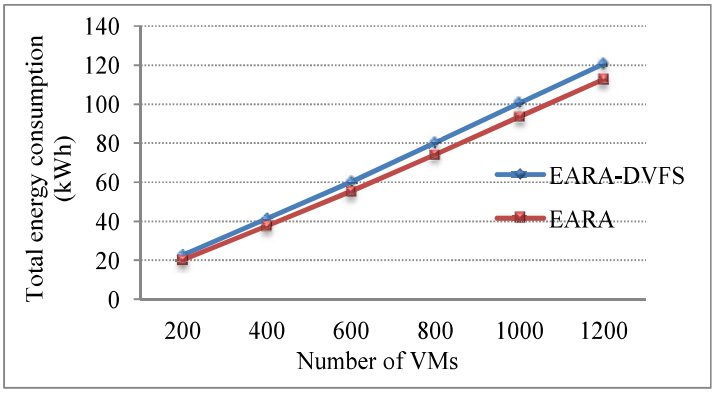


Figure 4.12: Comparison of total energy consumption between EARA and EARA-DVFS

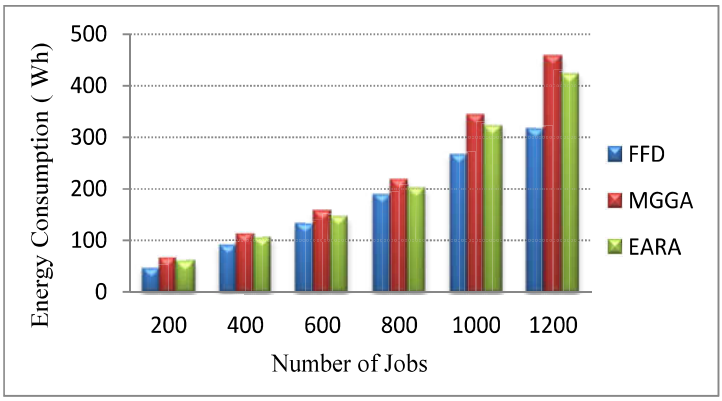


Figure 4.13: Comparison of computational energy consumption in FFD, MGGA, and EARA

and MGGA. This is due to the fact that EARA gives more weightage to resource availability when $\beta 2$ is set to 0. As all the data centers have exactly same computing infrastructure, so EARA distributes almost equal workload among them. This feature of EARA can be exploited for load balancing among datacenters.

Figure 4.12 shows a comparison of total energy consumption by the cloud computing infrastructure between EARA and EARA-DVFS. In case of EARA-DVFS, dynamic voltage frequency scaling is not applied. The result shows that EARA which is employing DVFS saves 8.15% more energy than EARA-DVFS.

Figure 4.13 depicts the comparison of computational energy for FFD, MGGA, and EARA. It is the total energy consumed for finding suitable resources for all the jobs. This figure shows that the EARA consumes less energy in computation than MGGA but little more than FFD. On an average, EARA consumes 0.42% of total energy consumption on the efficient allocation of resources. Therefore, EARA is better than FFD and MGGA as the computational energy consumption of 0.42% is very small compared to overall energy gain of 10.56%.

4.4 Summary

In this chapter, energy aware resource allocation methodology using ant colony optimization has been proposed. ACO is applied at two levels for efficient allocation of resources. First level ACO allocates VMs resources to jobs, whereas second level ACO allocates PMs resources to VMs. Resources are allocated to jobs on the basis of their

QoS requirements. Server consolidation and dynamic performance scaling of PMs are employed to conserve energy. The proposed methodology is implemented in CloudSim and the results are compared with FFD and MGGA resource allocation methods. It is experimentally established that proposed EARA achieves up to 10.56% saving in energy consumption through better utilization of resources.

The next chapter discusses about energy and QoS aware resource allocation.

Chapter 5

EQUAL: Energy and Quality of Service Aware Resource Allocation[3]

The previous chapter discussed energy efficient allocation of resources employing Ant Colony optimization at two levels. In addition to the huge energy consumption of the data centers, liability of fulfilling Quality of Service (QoS) requirements of the end users have made resource allocation a more challenging task.

In this chapter, energy and QoS aware resource allocation approach which employs Antlion optimization for allocation of resources to Virtual Machines (VMs) is proposed. It can operate in three modes, namely power aware, performance aware, and balanced mode. The proposed approach enhances energy efficiency of the cloud infrastructure by improving the utilization of resources while fulfilling QoS requirements of the end users.

This chapter starts with discussion about energy and QoS aware resource allocation approach. It then elaborates a power model, problem definition, and application and infrastructure model. Finally, it illustrates the Antlion optimization based resource allocation, and then concludes with the performance evaluation and comparative analysis of the proposed approach.

5.1 Energy and QoS Aware Resource Allocation

Resource allocation is a process of provisioning resources to VMs. Resources are allocated to VMs with the aim to minimize energy consumption while satisfying QoS

[3]Ashok Kumar, Rajesh Kumar, Anju Sharma, *"EQUAL: Energy and QoS Aware Resource Allocation Approach for Clouds"*, Computing and Informatics, ISSN 1335-9150, 2016 [in press]

requirements. The Antlion optimization is used for energy and QoS aware allocation of resources to VMs. The proposed approach can be operated in power, performance, and balanced mode. In power aware mode, a VM is allocated to a resource that causes minimum increase in energy consumption. Whereas in performance mode, a VM is allocated to a resource that has a maximum available computational capacity. In balanced mode, power and performance are given equal weightage while allocating resources. The VMs encapsulate users' tasks which are scheduled in Earliest Deadline First (EDF) order. Each task has a deadline, a point of time, by which execution of the task should finish. If deadline of a task is missed then SLA violation is said to have occurred.

In the proposed approach, the following assumptions are taken into consideration:

(i). Each task is independent of other tasks.

(ii). A VM can be executed on a server with lesser free available resources than required, but at the cost of reduced performance.

(iii). Resources can be switched to sleep mode to conserve energy.

(iv). Energy consumption of a resource in sleep mode is negligible.

The major components of energy and QoS aware resource allocation approach are shown in Figure 5.1. The *bag of tasks* is the collection of time constrained tasks submitted by the end users. The detailed information about each resource like type of resource and its computational capability is provided by *resource description* component. The *resource allocation* component refers *resource description* component while allocating resources to VMs. The resources are allocated to VMs employing Antlion optimization. Once the resources are provisioned to VMs, *resource scheduler* manages the scheduling of VMs on the provisioned resources. Utilization of each resource (server) is monitored at regular interval of time, and saved in *QoS metric database*. Resource utilization information is used by *migration manager* to perform server consolidation. It is invoked after a fixed interval of time. A VM is selected for migration based on the Interquartile Range (IQR) [202] of the utilization history data. VM migration is performed in two cases. First, when a resource is under-loaded, i.e. the

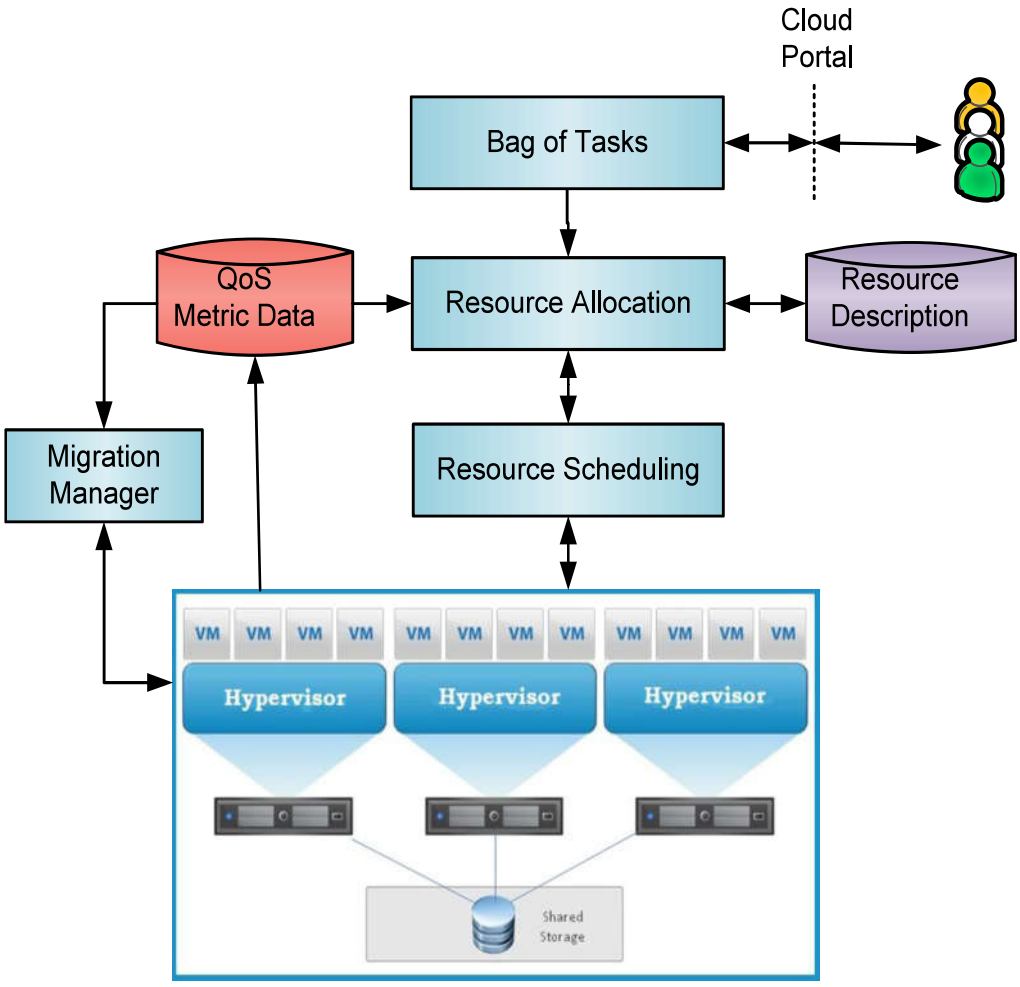


Figure 5.1: Energy and QoS Aware Resource Allocation

utilization of the resource is below Lower Green Threshold (LGT) limit. In this case, all the VMs running on the under-loaded resource are shifted to other resources and the resulted idle resource is switched to low power (sleep) mode to conserve energy. Second, when the resource is over-loaded, i.e. the utilization of the resource is above Upper Green Threshold (UGT). In this case, one of the VMs running over the resource is migrated to another resource to bring the resource utilization below UGT.

5.1.1 Motivation

Resource allocation in cloud computing can be accomplished using either traditional deterministic algorithms [49, 64, 123, 129, 130, 197] or metaheuristic algorithms [68, 70,

100–102, 110–112, 198, 199]. Deterministic algorithms suffer from local optima entrapment i.e. they got struck in local solutions and consequently fail to find the true global optimal solution. Moreover, resource allocation using deterministic algorithms is NP-hard. So, in the recent years metaheuristic algorithms have been employed for various fields in general and efficient allocation of resources in particular [68, 70, 100–112, 200]. The fundamental characteristic of metaheuristic algorithms is stochastic operators which are used for finding optimal solutions in the search space. Stochastic operators help them to escape local solutions. Due to their random behavior, they are able to obtain different solutions in each run. They start with some random solutions, called candidate solutions, of the problem at hand, and then improve the candidate solutions iteratively. Their solution finding process is completely independent of the problem. When a metaheuristic algorithm gets trapped in local solution, stochastic operators make random changes in the solution and eventually help in escaping from local optimal solution. In a nutshell, all metaheuristic algorithms follow a general and common framework, in which they improve a set of randomly created solutions iteratively. These algorithms differ in the method of improving the initial random solutions. The Antlion optimization over other existing metaheuristic algorithms used [68, 70, 100–112] for resource allocation because it provides very competitive results in terms of improved exploration, local optima avoidance, exploitation, and convergence [201]. Further, the proposed energy and QoS aware resource allocation approach significantly differs from the other metaheuristic based resource allocation approaches in the area of cloud computing as it can be tuned to operate in power aware, performance aware, or balanced mode.

The detailed working of proposed resource allocation approach, EQUAL, is presented in the next section.

5.1.2 Power Model

Generally, the resources have different run-time power consumption because of their heterogeneous processor architectures, processing speeds, hardware features, etc. Power

consumption of a resource is given by equation (5.1):

$$P_{total} = P_{dynamic} + P_{static}, \tag{5.1}$$

Static Power Consumption (SPC), P_{static}, is due to leakage current and is independent of clock frequency and usage scenario. SPC can be reduced by switching idle resources to sleep mode [71]. However, Dynamic Power Consumption (DPC), $P_{dynamic}$, is due to circuit activity, and it depends on resource utilization. DPC of a resource increases linearly with its utilization [94], and is given by equation (5.2).

$$P = P_{idle} + (P_{max} - P_{idle})U, \tag{5.2}$$

where P_{idle} is the power consumption when the resource is idle, P_{max} is the power consumption at 100% utilization, and P is the power consumption of the resource at utilization $U \in [0, 1]$.

5.1.3 Problem Definition

Cloud computing leverages virtualization, such as XEN [203], KVM [204], or VMWare [205] to support the execution of multiple VMs on a single physical resource. Each VM has some resource demands such as CPU, number of processing cores, memory, network bandwidth, etc. If a VM is not allocated required resource capacity, then it processes encapsulated tasks at slower speeds thereby elongating the task completion time. Consequently, some of the tasks may miss the deadline. When deadline of a task is not observed, it is considered as SLA violation.

Suppose a set $J = \{J_i | 1 \leq i \leq n\}$ of n tasks, and each task J_i is associated with deadline time d_i and processing volume w_i. The processing volume is the amount of processing in Millions of Instructions (MI) that must be carried out to finish the task. Tasks are distributed among a V number of VMs. Further, $S = \{S_j | 1 \leq j \leq m\}$ is a set of m resources. The problem is to allocate resources to VMs in such a way to minimize the number of active resources and their energy consumption while observing QoS requirements (deadline) of the end user's task. We considered only processing requirements while allocating resources because the CPU is accounted for the major

part of energy consumption by a physical machine [71]. The allocation of resources to VMs is *NP*-hard. Therefore, Antlion metaheuristic optimization is employed for allocation of resources to VMs. The proposed approach groups VMs over a small number of resources and thus allows turning off those resources that are not in use. Energy efficiency and QoS are considered while allocating resources to the VMs. The suitability of resource r for VM j is determined from fitness function $f_{j,r}$, given by equation (5.3), which helps in fulfilling the following goals:

(i). Allocation of a VM to a resource that results in minimum increase in energy consumption of the cloud.

(ii). Provisioning VMs on reduced number of resources.

(iii). Performance requirements are taken into consideration while allocating resources.

$$f_{j,r} = \frac{\left[\frac{\Re_r^a}{\Re_j^d}\right]^{\theta}}{\left[\gamma \triangle E_{j,r} + (1-\gamma)\kappa_r \underbrace{\left(1 - \sum_{i \in S, i \neq j} \Re_{i,r}\right)}_{y}\right]^{1-\theta}}, \qquad \forall r \tag{5.3}$$

where $\Re_r^a$ is available processing power of resource r, $\Re_j^d$ is processing demand of VM j. $\triangle E_{j,r}$ is the energy contribution of VM j on resource r, κ_r is energy affinity, $\Re_{i,r}$ is the fraction of processing power allocated to VM i on resource r, and $0 \leq \gamma \leq 1$ is a constant. $0 \leq \theta \leq 1$ is a trade off between performance and energy. By changing the value of θ, EQUAL can be operated in one of the three modes, namely: (i) Power aware, (ii) Performance aware, or (iii) Balanced, mode. EQUAL operates in power aware mode when θ is set to 0. In this mode, the increase in energy consumption of the resource is considered while allocating VMs. Thus, a VM is allocated to a resource which results in minimum increase in energy consumption. When θ is set to 1, EQUAL operates in performance aware mode, and thus allocates a VM to a resource which has a maximum available computing power at disposal. In balanced mode, when θ is 0.5, EQUAL maintains a balance between power and performance while allocating resources to VMs. EQUAL inclines towards power aware allocation if $\theta < 0.5$, and towards performance aware allocation if $\theta > 0.5$.

The ratio $\frac{\Re_r^a}{\Re_j^d}$ is called performance affinity which is desired to be greater than or equal to 1. When the performance affinity value is less than one, the VM would not get sufficient resource and therefore the execution of encapsulated tasks slows down. Energy affinity, κ_r, is the minimum energy consumption of the resource, i.e. energy consumption in idle state. Therefore, EQUAL gives preference to resources having lesser energy consumption in idle state. A resource having low power consumption in the idle state has higher value of fitness function and is therefore given preference over others while resource provisioning. The term y allows to group VMs on a lesser number of resources. The value of term y for a resource r decreases as more and more VMs are deployed on it. Consequently, fitness function of the resource r increases and thereby enables grouping of VMs on a lesser number of resources.

When θ is set to 1, power consumption of a resource does not contribute in finding suitable resource for the candidate VM. The machine having the maximum available resource capacity is given preference over the others and the allocation approach reduces to worst-fit decreasing. In order to consider power affinity while searching for best resource and to pack VMs on a lesser number of machines, the denominator of the equation (5.3) should not evaluate to 1. This is possible only if θ is assigned value smaller than one. Therefor, $\theta = 0.95$ is used to bias EQUAL towards performance aware allocation while taking advantage of its VM packing capability in order to save power consumption. However, when θ is set to 0, EQUAL reduces to best-fit approach and strives to pack VMs on a lesser number of resources. The machine which is hosting more VMs and has a low energy affinity is given preference over the others. The processing power of the resources is not taken into consideration. In order to consider the computing capability of the resource in power aware mode θ should be assigned a small positive value other than 0. Hence, $\theta = 0.05$ is used for power aware allocation in order to consider the computing power of a resource in addition to its power consumption.

5.1.4 Application and Infrastructure Model

Cloud computing is a suitable platform for deadline constrained scientific applications in areas such as astronomy, bioinformatics, and physics [206]. The proposed

resource allocation approach, EQUAL, can be used for deadline constrained applications such as Montage, which is used for generation of sky mosaics; Cyber-Shake, used for earthquake risk characterization; LIGO, used for detection of gravitational waves and SIPHT, used in bioinformatics. All these four applications are characterized by Juve *et al.* [207]. Scientific application (task) consists of thousands of sub-tasks, and can take benefit of large-scale infrastructure of cloud computing. Scientific application has a soft deadline, which is required to be accomplished. A soft-deadline does not make the computation useless if the task is not completed in time [208]. A computation begets maximum benefit if the deadline is achieved. A scientific application may consist of sub-tasks and may have dependencies between them. Each task i has a deadline d_i and processing volume w_i associated with it. Deadline of a task determines the time to accomplish the execution of the task from the moment it is submitted to EQUAL, which manages the execution of tasks, allocates VMs to them, and schedule their execution in the cloud. EQUAL offers a set of four VM types denoted by set $V = \{A0, A1, A2, A3\}$. Each VM has heterogeneous resource dimensions. There is no limit on the number of VMs of each type that can be running at any moment for the execution of tasks. The problem addressed in this chapter is the execution of tasks latest by deadline time at the smallest possible energy cost. The problem is solved by the efficient allocation of resources using Antlion optimization.

5.1.5 Antlion Optimization

Antlion Optimization [201] is proposed by Seyedali Mirjalili in 2015. Antlions belong to the myrmeleontidae family. An Antlion larvae makes cone-shaped pit and hides itself underneath the pit waiting for prey to be trapped in. The size of the pit is proportional to level of hunger. When an insect is trapped in the pit, the Antlion tries to catch it by intelligently throwing sand towards to edge of the pit to slide the prey into the bottom of the pit. Once the insect is caught, it is pulled under the sand and then consumed.

The following are the reasons for selecting Antlion optimization for resource allocation:

(i). Random selection of Antlions guarantees exploration of the search space.

(ii). Adaptive shrinking boundaries of Antlions' traps guarantee exploitation of search space.

(iii). The promising regions of the search space are guided by Antlions.

(iv). It is a gradient-free algorithm and considers the problem as a black box.

5.2 Resource Allocation using Antlion Optimization

The Antlion optimization algorithm, which mimics the behavior of Antlions and Ants, is used for discovering resource for a VM. The objective is to find a resource that fulfills not only the resource requirements of the VM but also causes a minimum increase in energy consumption. Each Antlion provides an initial guess of the resource, and then a resource better than the initial guess is searched through a random walk of an Ant around the Antlion. When a better resource is found, the location of the Antlion is replaced with the location of the corresponding Ant.

The location of the Ant and Antlion, each representing a resource, are saved in matrix M^a and M^{al}, respectively. Location of i^{th} Ant, W_i^t, and j^{th} Antlion, V_j^t, at t^{th} iteration is represented by i^{th} and j^{th} rows of matrices M^a and M^{al}, respectively. In each iteration, location of an Ant is updated to reflect its latest position. The fitness value of an Ant, which determines the goodness of a solution, is also updated in each iteration. The fitness value of an Ant/Antlion is evaluated from equation (5.3). When the fitness value of an Ant becomes greater than the fitness value of the Antlion, the location and fitness value of the Antlion are replaced with the location and fitness value of the corresponding Ant. The fitness values of Ants and Antlions are saved in matrix M^{fa} and M^{fal}, respectively.

Random walk of an Ant i in the search space is modeled by Levy Flight (LF) [209], which can be expressed by equation (5.4).

$$W_i^t = W_i^{t-1} + \alpha \; L(s, \lambda), \tag{5.4}$$

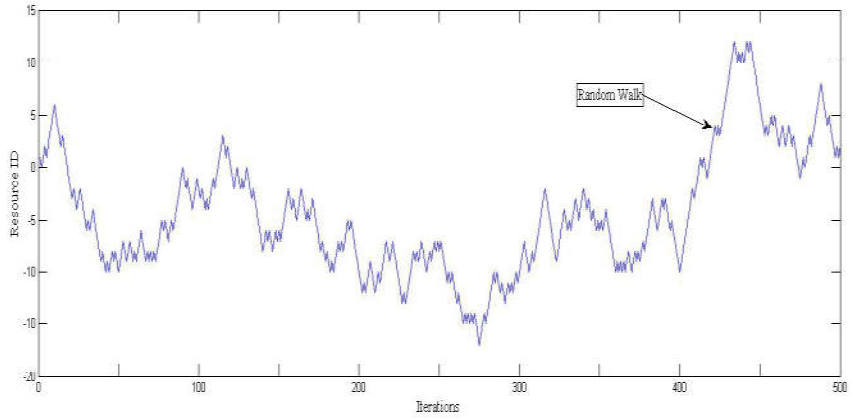


Figure 5.2: Random walk of an Ant

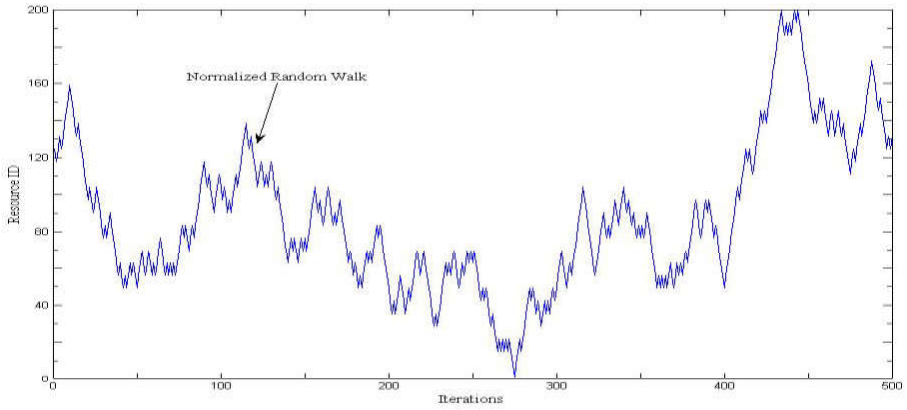


Figure 5.3: Normalized Random Walk of an Ant

where α is the scaling factor for step size s. Levy exponent, λ, is a constant. $L(s, \lambda)$ is levy distribution with parameters s and λ. W_i^t is the location of an Ant i at t^{th} iteration. Figure 5.2 shows random walk for an Ant generated using equation (5.4). Number of iterations is represented along x-axis, whereas, resource ID (identification) is denoted by y-axis. Since search space, consisting of identification of each resource, has a range of permitted values, so max–min normalization, as shown in equation (5.5), is applied to keep the random walk within the desired range.

$$W_i^t = \left\lceil \frac{(W_i^t - a_i)(d_i^t - c_i^t)}{b_i - a_i} + c_i^t \right\rceil \tag{5.5}$$

where a_i and b_i are the minimum and maximum of random walk of i^{th} Ant, and c_i^t, d_i^t are the minimum and maximum of search space at t^{th} iteration. Figure 5.3 shows normalized random walk of the Ant generated using equation (5.5). Random walk of an Ant (shown in Figure 5.2) is normalized to a range of resource identifications used

in EQUAL i.e. from 1 to 200.

Random walks of Ants are affected by the positions of Antlions. An Ant is allowed to move around an Antlion which is selected using roulette wheel. The range of search space at t^{th} iteration for random walk of an Ant i around the Antlion j is mathematically modeled by equations (5.6) and (5.7).

$$c_i^t = V_j^t + c^t, \tag{5.6}$$

$$d_i^t = V_j^t + d^t, \tag{5.7}$$

where c^t and d^t are the minimum and maximum of the search space at t^{th} iteration, c_i^t and d_i^t are the minimum and maximum for i^{th} Ant, and V_j^t is the position of the selected j^{th} Antlion at t^{th} iteration. In order to find the best resource, values of c^t and d^t are updated in each iteration using equations (5.8) and (5.9), respectively.

$$c^t = \left\lceil \frac{c^t}{I} \right\rceil, \tag{5.8}$$

$$d^t = \left\lceil \frac{d^t}{I} \right\rceil, \tag{5.9}$$

here, $I = 10^w \frac{t}{T}$, where t is the current iteration, T is the maximum number of iterations, and w is a constant that can adjust the level of exploitation and is defined on the basis of the current iteration.

As discussed earlier, the location of an Antlion is replaced with the location of corresponding Ant when the fitness of a resource referred by an Ant becomes greater than the fitness of a resource referred by the Antlion. This situation is represented by equation (5.10).

$$V_j^t = W_i^t, \quad \text{if } f(W_i^t) > f(V_j^t) \tag{5.10}$$

where V_j^t is location of j^{th} Antlion at t^{th} iteration, and W_i^t is location of i^{th} Ant at t^{th} iteration.

EQUAL maintains a record of the best resource (solution). The best resource, called elite, is saved in each iteration. The elite solution has the highest fitness value and

effects the random walk of each Ant (as shown in equation. (5.11)).

$$W_i^t = \frac{W_i^t + W_e^t}{2} \tag{5.11}$$

where W_i^t is a random walk of Ant i around an Antlion and W_e^t is a random walk of elite e at t^{th} iteration.

The detailed resource provisioning process is given in Algorithm 5.1. It employs Antlion optimization to find the best resource for a VM. The resource search process is repeated until maximum iterations T has elapsed or the elite solution is same for three consecutive iterations.

Algorithm 5.1 Pseudo code for energy and QoS aware resource allocation using Antlion Optimization

Input: Set V of VMs; Set A of Ants; Set L of Antlions; Set S of resources; Maximum iterations T

Output: VMs-Resources map (M_{VR})

for each VM $v \in V$ **do**

Initialize Ants' position matrix M^a randomly.

Initialize Antlions' position matrix M^{al} randomly.

Evaluate suitability of resource referred by each Ant $i \in A$ ($f_{v,i}$) and Antlion $j \in L$ ($f_{v,j}$) for VM v from fitness function (equation (5.3)) and store the values at i^{th} and j^{th} row of matrix M^{fa} and M^{fal}, respectively.

Find an Antlion (say e) for which value of fitness function (equation (5.3)) is maximum (say $f_{v,e}$) and call it elite solution.

set iteration counter $t \leftarrow 1$

while ($t \leq$ T) **and** (until e is same for three consecutive iterations) **do**

for each Ant $i \in A$ **do**

Select an Antlion j using Roulette wheel.

Calculate c^t and d^t using equations (5.8) and (5.9).

Evaluate c_i^t and d_i^t using equations (5.6) and (5.7) to select range of random walk for Ant i

Generate random walk for Ant i using equation (5.4)

Normalize random walk for Ant i using equation (5.5)

Update random walk on Ant i using equation (5.11).

Update position vector (M_i^a) and fitness value (M_i^{fa}) of Ant i.

if ($f_{v,i} > f_{v,j}$) **then**

set $M_j^{fal} = M_i^{fa}$

set $M_j^{al} = M_i^a$

end if

end for

Find new elite solution among Antlions and assign it to e.

set $t \leftarrow t + 1$

end while

Allocate VM v to resource referred by elite solution e, and add VM-resource pair to map M_{VR}

end for

return M_{VR}

5.3 Performance evaluation and comparative analysis

A number of cloud simulation tools such as CloudSim [176], CloudAnalyst [210], GreenCloud [211], NetworkCloudSim [212], etc. are available to implement and evaluate a resource allocation approach on a large scale, repeatable, and controlled cloud environment. But the proposed approach is implemented in CloudSim because it supports modeling of various cloud entities such as data centers, servers, virtual machines, and tasks with ease. The proposed resource allocation approach can be implemented easily by extending VM allocation policy of CloudSim.

For performance analysis, EQUAL is compared with Artificial Bee Colony (ABC) [102],

Table 5.1: Specification of Resources

Type	*Processing*	*PEs*	*RAM*	*Storage*	*BW*
1	2933	4	8	500	10
2	3067	4	8	500	10
3	2933	12	12	500	10
3	3067	12	16	500	10

Processing, processing speed in millions of instructions per second; PEs, number of processing elements; RAM, random access memory in GB; Storage, Permanent storage capacity in GB; BW, network bandwidth in gigabits per second.

Table 5.2: Specification of Virtual Machines

VM Type	*CPU*	*PEs*	*RAM*	*BW*
A0	500	1	768	1000
A1	1000	1	1792	1000
A2	1500	2	3584	1000
A3	2000	4	7168	1000

CPU, processing speed in millions of instructions per second; PEs, number of cores; RAM, random access memory in mega bytes; BW, network bandwidth in megabits per second.

Table 5.3: Simulation Parameters

Number of Resources	50-200	
Number of Tasks (Cloudlets)	200-1000	Varied in every simulation run
Size of tasks	10000+(5-30%)MI	in millions of instructions(MI)
Simulation Span	86400s	Simulation time period
Idle Time	10 min.	Time to switch PM to sleep mode
UGT	0.85	Upper Green Threshold limit
LGT	0.20	Lower Green Threshold limit
HT	0.95	Hot-spot Threshold
CT	0.15	Cold-spot Threshold

Genetic Algorithm (GA) [126], and Non-QoS aware Resource Allocation (NQRA) which is designed by combining round robin and earliest deadline first scheduling approach that allocates resources using best effort approach.

5.3.1 Experimental Setup

The simulation testbed consists of a data center containing 200 resources. The specification of four types of resources used in the simulation is as per Table 5.1. An equal number of resources of each type are created in a simulation run. The datacenter models instances of general purpose compute-basic tier of Microsoft Azure [213], and the parameters relevant for the experiments are shown in Table 5.2. The tasks having diverse CPU and memory requirements are used, and the number of tasks is varied from 200 to 1000. Further, the tasks are modeled as Cloudlets and their processing requirements are represented in MI. Simulation is repeated forty to fifty times with different number of resources, VMs, and tasks. The different parameters used during the simulation are shown in Table 5.3.

5.3.2 Simulation Results

Case 1: EQUAL in Balanced Mode

EQUAL switches to balanced mode when 0.5 is assigned to θ. In this mode, energy

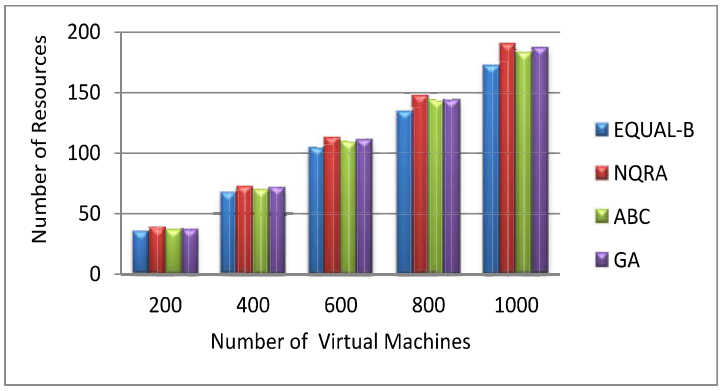


Figure 5.4: Comparison of the number of resources required

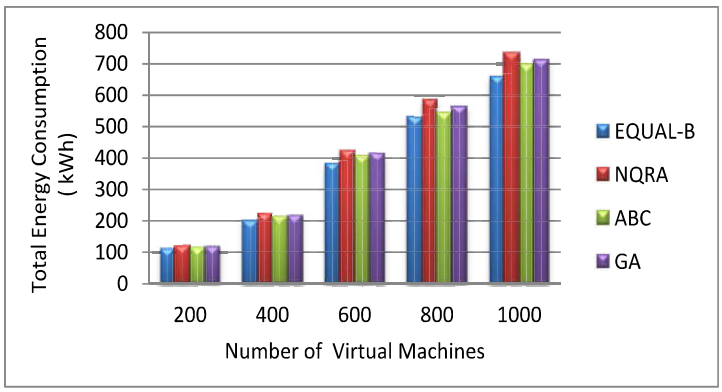


Figure 5.5: Comparison of Total Energy Consumption

and performance are given equal weightage while allocating resources to VMs. In order to group VMs over the minimum number of resources 0.05 is assigned to γ.

Figure 5.4 shows the comparison of a number of resources used by EQUAL in balanced mode (EQUAL-B), NQRA, ABC, and GA for different number of VMs. The number of used resources increases with increase in number of VMs to be deployed. But EQUAL-B uses a lesser number of resources than NQRA, ABC, and GA for given number of VMs. It has been observed that EQUAL-B uses approximately 8.68%, 4.47%, and 6.84% lesser number of resources than NQRA, ABC, and GA, respectively.

Figure 5.5 depicts the comparison of total energy consumption of EQUAL-B, NQRA, ABC, and GA. It is observed that EQUAL-B consumes lesser energy than NQRA, ABC, and GA for given number of VMs. In EQUAL-B, energy consumption of 107.67 kWh is measured for 200 VMs, and it increases to 735.65 kWh for 1000 VMs. It is observed from simulation results that EQUAL-B consumes 10.8%, 5.44%, and 7.69% lesser amount of energy than NQRA, ABC, and GA, respectively. Figure 5.6 narrates

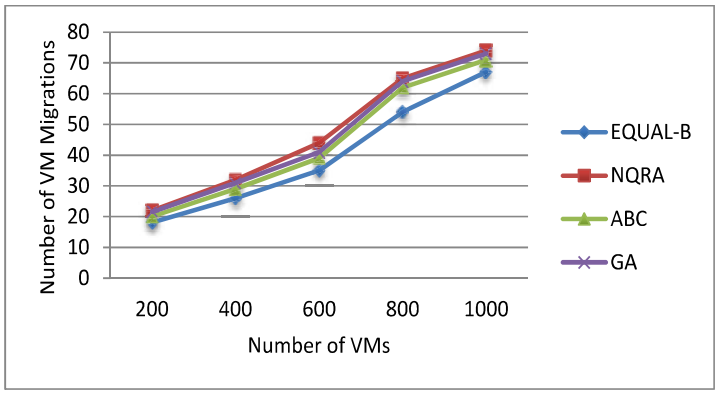


Figure 5.6: Comparison of number of VM Migrations

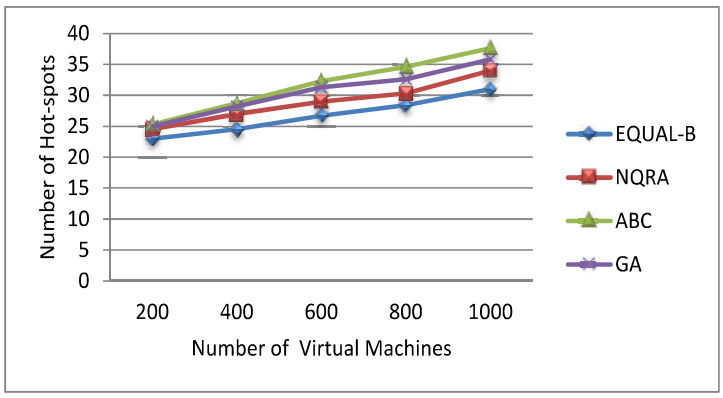


Figure 5.7: Comparison of number of Hot-spots

comparison of the average number of VM migrations performed in EQUAL-B, NQRA, ABC, and GA for different number of VMs. A VM is selected for migration using IQR. Resources becoming idle because of consolidation are switched to low power (sleep) mode to conserve energy. The number of VM migrations increases with the increase in number of VMs. In EQUAL-B, 16.75%, 10.97%, and 16.65% lesser number of VM migrations are observed than NQRA, ABC, and GA, respectively.

Figure 5.7 describes comparison of number of hot-spots created in EQUAL-B, NQRA, ABC, and GA. The number of VMs is varied from 200 VMs to 1000 VMs with increment of 200 VMs. A resource is considered as a hot-spot if its utilization is above HT. A hot-spot adversely affects the reliability and performance of the resource. Moreover, a hot-spot also demands better cooling arrangements. EQUAL-B improves reliability and energy efficiency of the resource as it creates 8.33%, 18.20%, and 14.22% lesser numbers of hot-spots than NQRA, ABC, and GA, respectively.

Figure 5.8 outlines the comparison of a number of cold-spots observed in EQUAL-B,

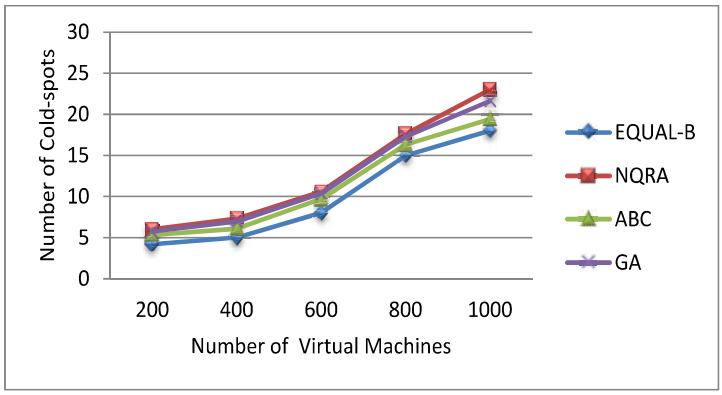


Figure 5.8: Comparison of number of Cold-spots

NQRA, ABC, and GA as the number of VMs are varied from 200 to 1000 VMs. A resource is considered as a cold-spot if its utilization is below CT. The number of cold-spots portrays the extent of resource wastage. EQUAL-B creates 18, whereas NQRA, ABC, and GA create 25, 19.41, and 21.6 average number of cold-spots when tested with 1000 VMs. The percentage of cold-spots generated in EQUAL-B, NQRA, ABC, and GA are 9.77%, 11.52%, 10.52%, and 11.25%, respectively. This shows that EQUAL-B manages the resources more efficiently.

Figure 5.9 outlines a number of deadlines missed in EQUAL-B, NQRA, ABC, and GA. A time constrained task executing in a VM is said to miss the deadline if it is not accomplished in stipulated time. The number of tasks is varied from 200 to 1000. In each run of the simulation 200 VMs are used. The tasks are distributed equally among the VMs. It is observed that a number of deadlines missed increases with the increase in number of tasks, but the rate of increase of missed deadlines is least in EQUAL. In EQUAL-B, 28.57%, 11.76%, and 25.00% lesser deadline misses are observed than NQRA, ABC and GA, when number of tasks are 200. However, for 1000 tasks, 21.58%, 9.57%, and 17.33% lesser tasks miss their deadline in EQUAL-B as compared to NQRA, ABC, and GA.

Figure 5.10 shows a comparison of allocation overhead that is total time taken by an algorithm to find the most suitable resource for each VMs. Allocation overhead of EQUAL is more than that of NQRA. However, it is lesser than ABC and GA. In case of EQUAL-B, average allocation overhead is about 26s for 200 VMs and it increases to 112s for 1000 VMs. However, the average allocation overhead of NQRA, ABC and GA for 200 VMs is 20s, 28.5s and 30s, and for 1000 VMs it is 92s, 122s and 128s,

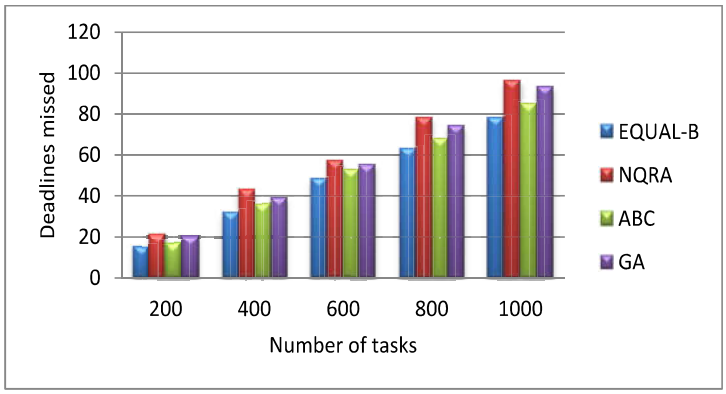


Figure 5.9: Comparison of number of Deadlines missed

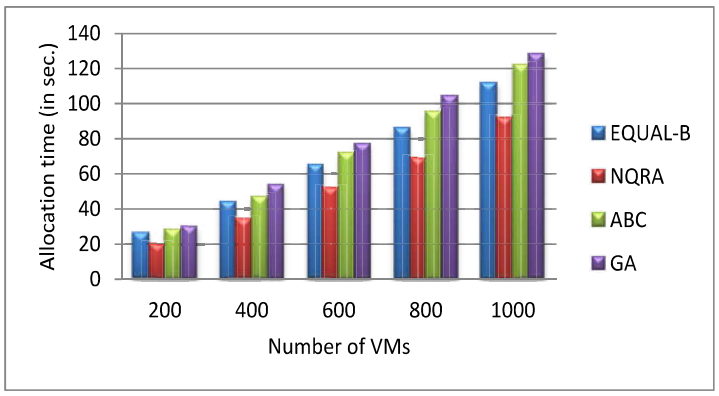


Figure 5.10: Comparison of Total Allocation Time

respectively. The results indicate that EQUAL-B has a higher convergence rate than ABC and GA.

Case 2: EQUAL in Energy-Aware Mode

The variable θ is assigned value 0.05 to operate EQUAL in energy aware mode (EQUAL-E). In this mode of operation, energy consumption of resources is considered while allocating resources to VMs. A VM is allocated to a resource that results in minimum increase in energy consumption. Further, the control variable γ is given the value 0.05 to pack VMs on minimum number of resources.

Figure 5.11 depicts the comparison of a number of resources used by EQUAL-E, NQRA, ABC, and GA for different number of VMs. The results show that EQUAL-E uses a lesser number of resources than NQRA, ABC, and GA for a given number of VMs. It is observed that EQUAL-E uses 14.47%, 10.05%, and 12.54 % lesser number of resources than EQRA, ABC, and GA, respectively.

Figure 5.12 outlines the comparison of energy consumption of EQUAL-E, NQRA,

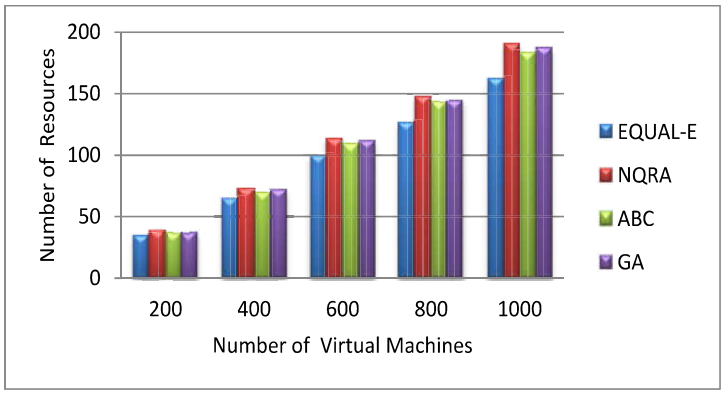


Figure 5.11: Comparison of the number of resources required

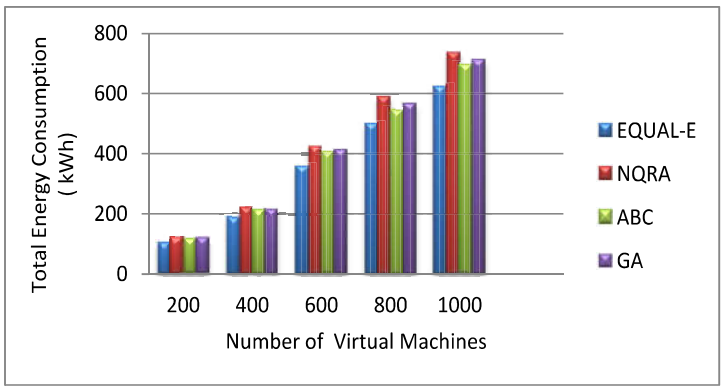


Figure 5.12: Comparison of total energy consumption

ABC, and GA. EQUAL-E saves energy by packing VMs on a lesser number of resources. As compared to NQRA, ABC, and GA average energy savings of 15.04%, 11.91%, and 14.30% is observed in EQUAL-E.

Figure 5.13 sketches comparison of the average number of VM migrations performed in EQUAL-E, NQRA, ABC, and GA. VMs are migrated from either under-loaded or over-loaded resources. A resource is considered under-loaded if its utilization is below LGT and over-loaded if its utilization is above UGT. It is observed that number of VM migration increases as the number of VMs have increased from 200 to 1000. In EQUAL-E, 9.37% 3.45%, and 6.05% lesser number of VM migrations are observed than NQRA, ABC, and GA, respectively.

Figure 5.14 depicts the comparison of number of hot-spots created in EQUAL-E, NQRA, ABC, and GA as the number of VMs are changed from 200 to 1000 VMs. As compared to EQUAL-B, EQUAL-E packs VMs on a lesser number of resources. As a result, number of hot-spots increases and gap of percentage number of hot-spots

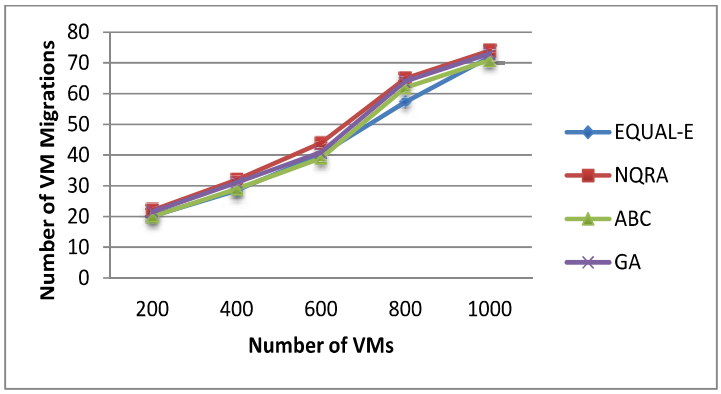


Figure 5.13: Comparison of number of VM migrations

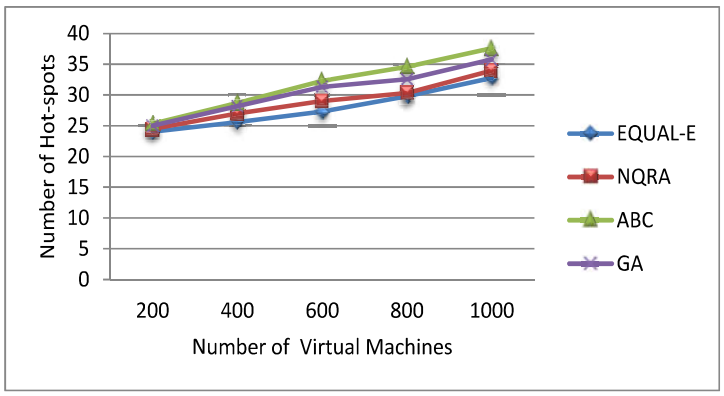


Figure 5.14: Comparison of number of Hot-spots

between EQUAL-E and NQRA, ABC, and GA reduce to 3.86%, 13.33%, and 9.52%, respectively.

Figure 5.15 shows the comparison of a number of cold-spots observed in EQUAL-E, NQRA, ABC, and GA. In EQUAL-E, fewer number of cold-spots are observed than EQUAL-B. On an average, approximately 16 cold-spots are observed in EQUAL-E against 1000 VMs compared to 23, 19.41, and 21.6 average number of cold-spots in NQRA, ABC, and GA for the same number of VMs.

Figure 5.16 narrates comparison of the number of tasks missed their deadline in EQUAL-E, NQRA, ABC, and GA. The number of tasks is changed from 200 to 1000. In each simulation run 200 VMs are used. Further, the tasks are distributed equally among the VMs. In EQUAL-E, VMs are mapped on a lesser number of resources than EQUAL-B. As a result, tasks encapsulated in the VMs do not get sufficient resources, causing an increase in number of deadline miss. Due to this, percentage deadlines missed gap between EQUAL-E and the other three approaches, i.e. NQRA, ABC, and

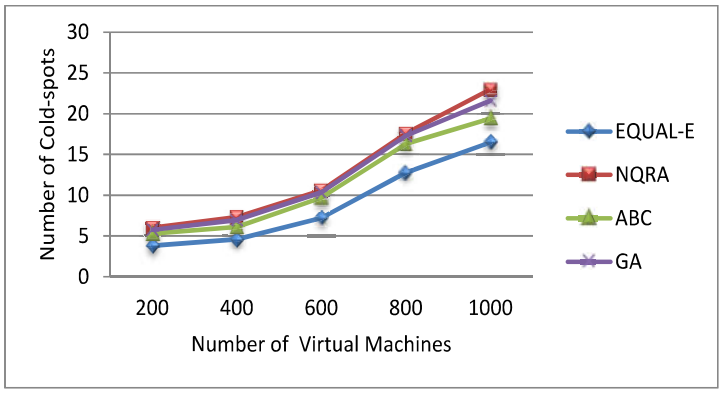


Figure 5.15: Comparison of number of Cold-spots

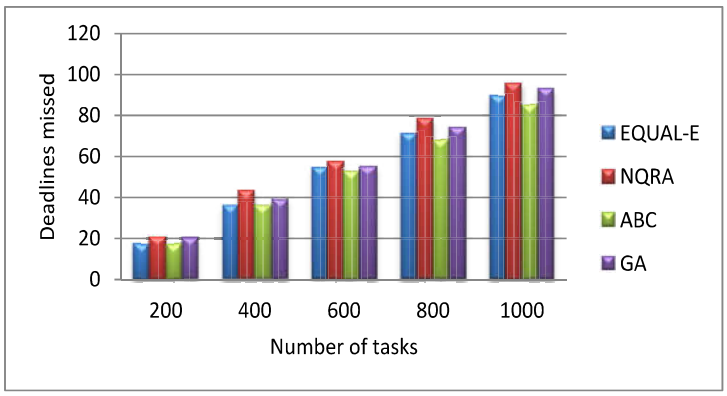


Figure 5.16: Comparison of the number of missed deadlines

GA reduces to 13.25%, 6.28%, and 7.31%, respectively

Case 3: EQUAL in Performance-Aware Mode

The variable θ is assigned value 0.95 to tune EQUAL to performance-aware mode (EQUAL-P). In this mode of operation, available computational capacity of each resource is considered while discovering suitable resource for a VM. Since the VMs are required to be allocated on minimum number of resources, so value 0.05 is assigned to control parameter γ.

Figure 5.17 shows the comparison of a number of resources used by EQUAL-P, NQRA, ABC, and GA. In EQUAL-P, a resource with higher performance affinity value is given preference over the others. In EQUAL-P more number of resources are used than EQUAL-B and EQUAL-E for given number of VMs. It is observed that EQUAL-E uses 8.20%, 4.98%, and 7.23% lesser number of resources than NQRA, ABC, and GA, respectively.

Figure 5.18 depicts the comparison of energy consumption of EQUAL-P, NQRA,

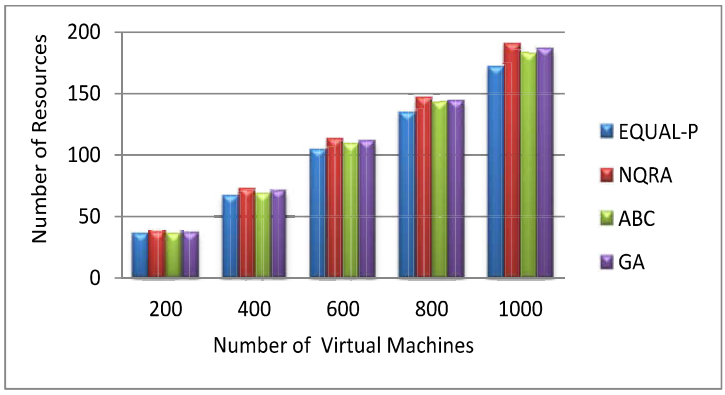


Figure 5.17: Comparison of the number of resources required

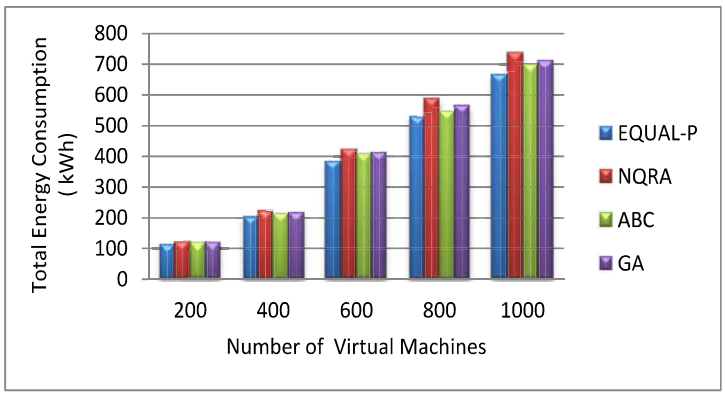


Figure 5.18: Comparison of total energy consumption

ABC, and GA. Energy consumption in EQUAL-P for given number of VMs is more than energy consumption in EQUAL-B and EQUAL-E because it uses larger number of resources. However, energy consumption in EQUAL-P is lesser than that of ABC and GA. Energy consumption of EQUAL-P is measured 8.77%, 4.73% and 6.94% lesser for 200 VMs, and 9.75%, 5.04% and 6.98% lesser for 1000 VMs than NQRA, ABC and GA, respectively.

Figure 5.19 represents a comparison of the average number of VM migrations performed in EQUAL-P, NQRA, ABC, and GA. In EQUAL-P, an average of 16.8 and 64 migrations are observed for 200 VMs and 1000 VMs, respectively. In EQUAL-P, number of migrations are lesser than the number of migrations in balanced and energy aware mode.

Figure 5.20 depicts the comparison of number of hot-spots created in EQUAL-P, NQRA, ABC, and GA. In EQUAL-P, fewer number of hot-spots than EQUAL-P and EQUAL-B are observed. It is observed that EQUAL-P creates 11.6%, 21.31%, and

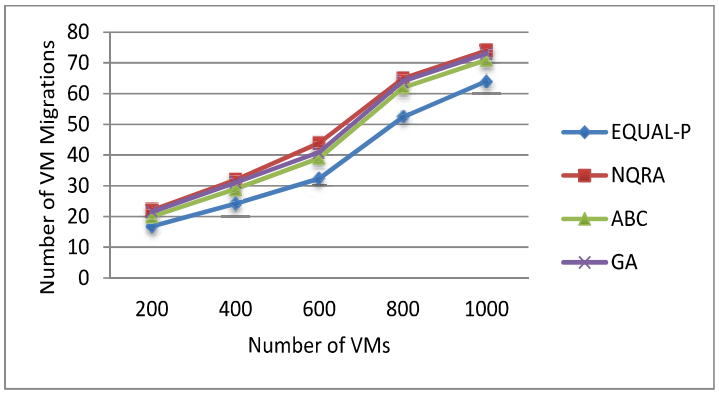


Figure 5.19: Comparison of number of VM migrations

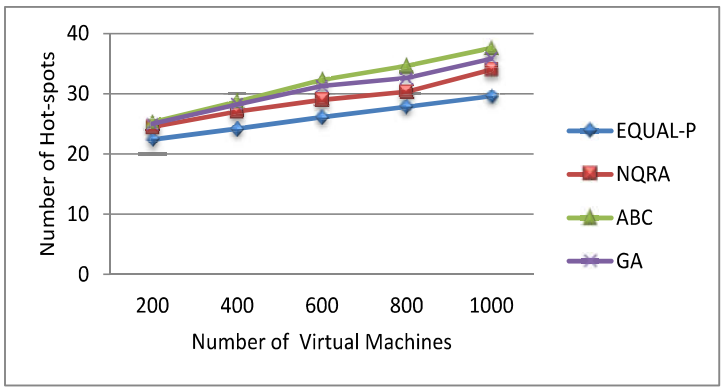


Figure 5.20: Comparison of number of Hot-spots

17.21% lesser number of hot-spots than NQRA, ABC, and GA, respectively.

Figure 5.21 shows the comparison of a number of cold-spots observed in EQUAL-P, NQRA, ABC, and GA. A resource is considered as a cold-spot if its utilization is below CT. A large proportion of the resource capacity goes waste if it is a cold-spot. Therefore, the larger is the number of cold-spots the greater is the resource wastage. It is observed that EQUAL-P generates 13.68%, 6.12%, and 11.66% lesser number of cold-spots than NQRA, ABC and GA. Therefore, it utilizes the resources more efficiently.

Figure 5.22 outlines a comparison of the average number of deadlines missed by EQUAL-P, NQRA, ABC, and GA. In each simulation run 200 VMs are used. Number of task are varied from 200 to 1000 and are distributed equally among the VMs. In EQUAL-P, fewer number of tasks miss their deadlines than EQUAL-B and EQUAL-E. This is due to the fact that, EQUAL-P uses more resources to map given number of VMs. In EQUAL-P, On an average, 12 tasks miss their deadlines when the total

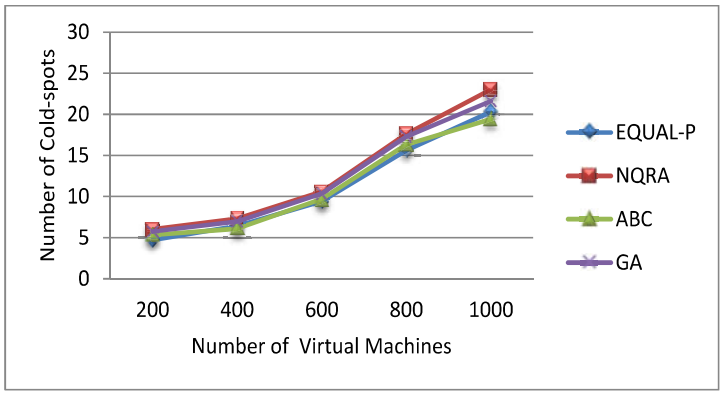


Figure 5.21: Comparison of number of Cold-spots

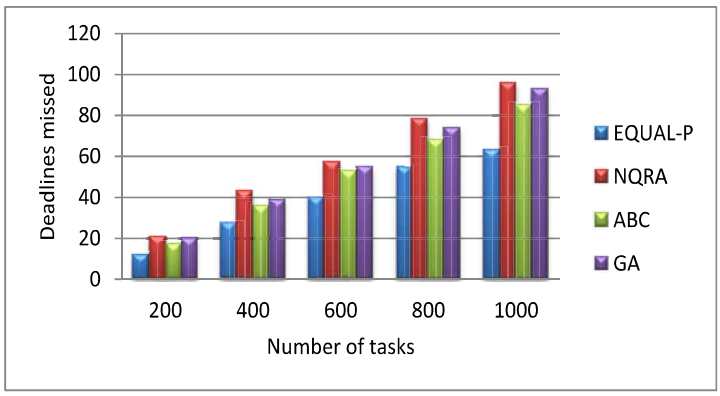


Figure 5.22: Comparison of the number of missed deadlines

number of tasks are 200, whereas 63 tasks miss the deadlines when the total number of tasks are increased to 1000. However for NQRA, ABC and GA, average number of tasks miss their deadline are 21, 17 and 20 for total 200 tasks, and 96, 85 and 93 for 1000 tasks.

5.4 Summary

In this chapter, energy and QoS aware resource allocation approach (EQUAL) is proposed. Antlion optimization is used for allocation of resources to VMs which encapsulate heterogeneous time constrained tasks. EQUAL can be governed to operate in one of the three modes, namely power aware, performance aware and balanced mode.

The proposed approach is implemented in CloudSim, and tested with VMs/tasks having diverse resource requirements. The experimental results prove that the proposed approach reduces the energy consumption up to 15%, and also improves quality of service in terms of reduction in percentage of tasks missed their deadlines.

The next chapter presents a self optimizing system for energy and QoS aware resource allocation for optimizing the allocation of resources in accordance with the workload of the applications.

Chapter 6

A Self Optimizing System for Energy and Quality of Service Aware Resource Allocation

The previous chapter presented energy and Quality of Service (QoS) aware resource allocation approach that can be tuned offline to operate in one of the three modes namely power, performance, and balanced mode. This chapter presents an extended version of energy and QoS aware resource allocation approach, presented in the previous chapter, to supports self-optimizing of resources allocated to the applications in accordance with their workload. The resource utilization history of the applications is used for predicting the resource needs of the applications in the near future. The proposed self optimizing system automatically switches between different modes of operations inline with the projected workload of the applications.

The chapter presents Self-Optimizing System (SOS) for energy and QoS aware resource allocation covering Monitoring, Analyze, Plan and Execute phase of this autonomic system. It then discusses criterion for server selection and server selection process for energy-efficient and QoS aware allocation of resources to the applications. The chapter further presents optimization of resource allocation using Antlion optimization. The chapter concludes with discussion of performance evaluation of the proposed self optimizing approach.

6.1 Self Optimizing System for Energy and QoS Aware Resource Allocation

To accommodate growing demand for computational resources, Cloud market players such as Amazon, Microsoft, Google, GoGrid, and Flexiant. have set up large sized data centers. These data centers consume huge amount of energy, and a major proportion of total energy consumption of these data centers gets wasted because of inefficient resource management. The problem is further aggravated by the growing size of data centers, heterogeneity of resources, variations in workload, and liability to satisfy QoS requirements. Thus, there is a pressing need of a autonomic resource allocation system that not only conforms to QoS requirements of end users but also conserves energy. The motivation for the proposed work stems from the challenges associated with resource allocation [52, 214]. The SOS can automatically switch between power-aware mode, performance-aware mode, or balanced mode depending on the intensity of workload predicted for the next hour. The mode switching is based on the outcomes of some pre-established rules. The aim behind proposed SOS for resource allocation is to enhance QoS and energy efficiency of the Cloud by autonomic adjustment in resources allocated to the virtual instances of the application in order to keep the number of physical machines (resources) to minimum and thus reducing carbon footprints.

The end users located at different geographical locations can access the applications round the clock and thus cause fluctuating resource demands at different times of the day. Each Cloud application is encapsulated in a VM, and leveraging virtualization, multiple applications can be multiplexed on single PM. These applications demand for some amount of processing, memory, network bandwidth, etc., and are associated with performance, cost and energy minimization constraints.

The service providers can be benefited by maximizing utilization of the resources by deploying applications on minimum number of servers while respecting their QoS demands. The end users are benefited by executing their applications in minimum time and cost. Further, the end users not only expect response time, cost, etc. to be minimized, but also want the applications to be available round the clock.

The proposed SOS has been designed keeping the interest of both the provider and the consumer in view. Figure 6.1 shows the framework of self-optimizing resource allocation system. The self-optimizing resource allocation process is divided into four

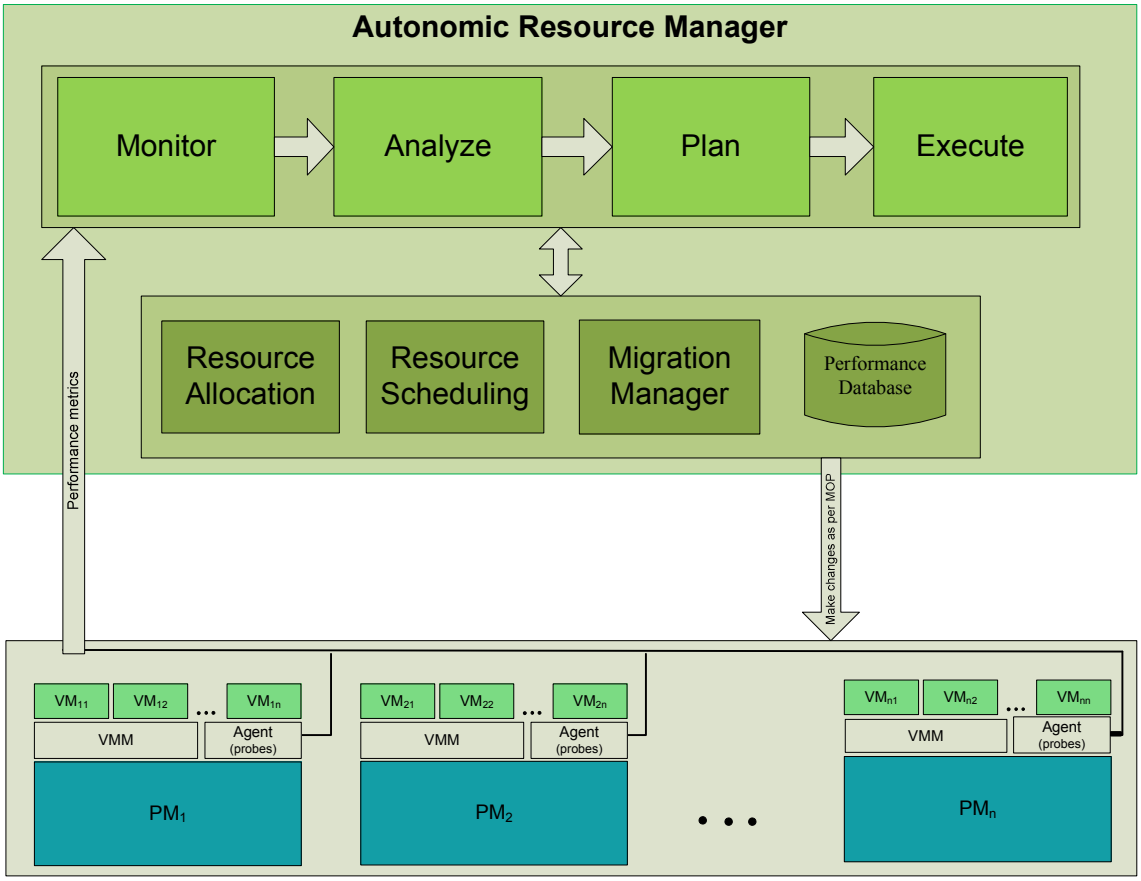


Figure 6.1: Energy and QoS aware resource allocation framework

phases:

(i) Monitor, (ii) Analyze, (iii) Plan, and (iv) Execute

Monitor: In this phase, performance metrics such as CPU and memory utilization of the hosts and the VMs are collected for further analysis. The metrics are collected and then sent to the monitoring module using push paradigm. These metrics are gathered using specialized programs, called probes, that can be easily incorporated into the system to get instantaneous values of utilization metrics. In the push model, each host periodically sends metrics data to the central controller using User Datagram Protocol (UDP). The collected performance-metrics data is saved in *performance database* for later processing. The SOS is highly scalable as data is collected on each physical machine and then sent to the central *Monitor* module using a lightweight communication

protocol. The push model is supported by Kernel-based Virtual Machine (KVM) and Xen, but VMWare [215].

Analyze: This module analyzes the CPU utilization data gathered by the monitoring module in order to infer mode of operation. When to switch and which mode to switch to are the critical issues to be addressed? The first question is addressed by applying prediction on past resource utilization history. The predictive approach infers resource utilization over time scales of hours. The peak demand of the application is predicted over the several hours of the day to decide the mode of operation. The use of prediction is motivated by long term variations in the web applications [216]. For instance, the workload of the retail internet application is higher during noon of every day. The cyclic pattern of the applications can be predicted ahead of time by analyzing past observations. The analyze module is based on the technique presented by Gmach *et al.* [217]. It uses the past observations of the applications to predict the peak demand that will be observed over a time period of T time units. The peak demand of the application is predicted for the next one hour, at the beginning of each hour. For this, utilization corresponding to a session arrival rate of the application is maintained. A web application observes seasonal overloads, the peak value of their resource demands often far more than their normal resource needs. *Plan:* The past values of resource utilization of an application play a key role in anticipating the mode of operation for SOS. The decision of switching to a different mode of operation is based on perspicacious analysis of the utilization data. The predicted utilization of the application for the next hour is used to determine the mode of operation as shown in Figure 6.2. The two threshold utilization values have been set to resolve the second issue, i.e. which mode to switch to? SOS switches to a different mode of operation when the predicted utilization of an application crosses either lower or upper threshold limit. SOS switches to power-aware mode when the application's workload is below lower threshold. In this situation, the virtual instances of the application are consolidated to lesser number of servers, and idle servers are switched to low power modes to conserve energy. Note that by virtue of elasticity, multiple instances of an application are created to accommodate the workload of the application. When the workload is very low, the resource demands of the applications can be met after consolidation to fewer number of servers. However, when the application's workload is high, the more

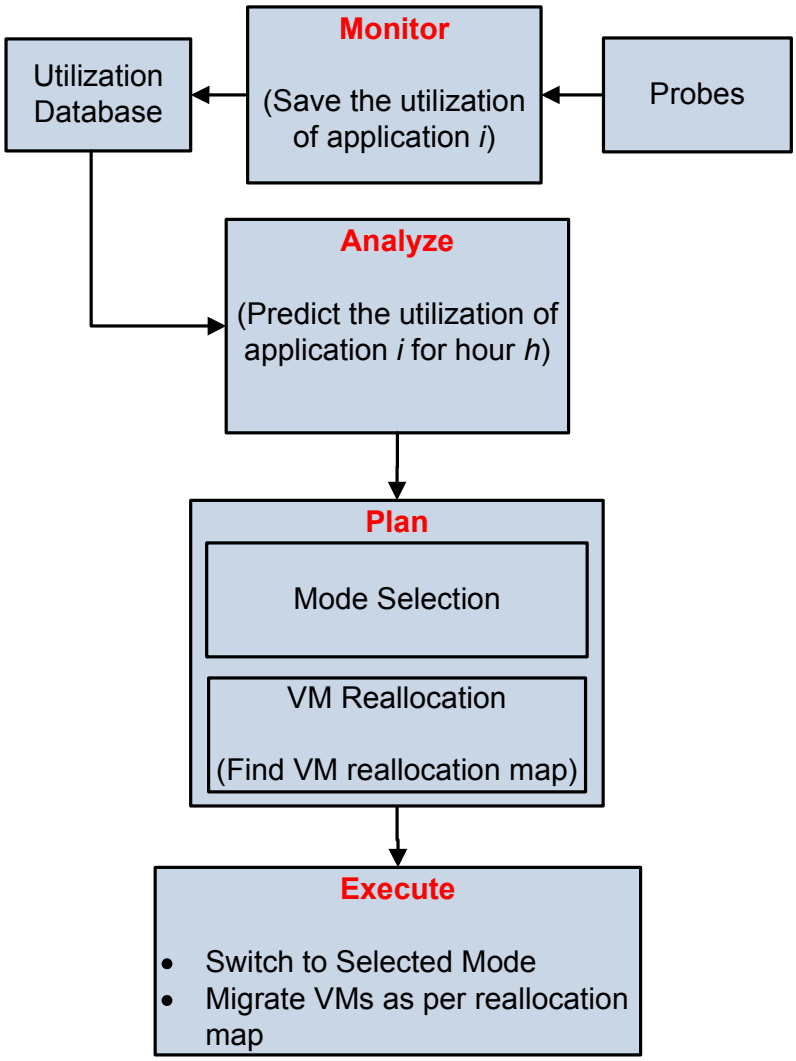


Figure 6.2: Selft-optimization of Resources

resources are needed to efficiently handle the users' request in compliance with QoS requirements. For instance, web applications generally have a response time as QoS parameter. Therefore, the service requests of such type of applications need to be completed in minimum time for customer satisfaction. In this case, the applications should be allocated enough resources to efficiently handle the users' requests. Therefore the applications should be allocated to a server that have enough free resource capacity to fulfill the resource demands of the application. Hence, SOS switches to performance aware mode to allocate resources to an application on a physical machine that have sufficient free resources to fulfill the resource demands of the application. In this mode of operation, performance of the application is given preference over energy

consumption. In balanced mode, both energy consumption and QoS are taken into consideration, the resources are allocated to minimize the energy consumption without sacrificing service level objectives. After deciding for the mode of operation, the plan phase evaluates the VMs which are required to be migrated. The VM migration is performed for two purposes: to consolidate the VMs on a lesser number of servers to conserve energy when the utilization is very low, and to handle the service requests in compliance with QoS by shifting the application from heavily loaded server to another server with enough spare source capacity to accommodate the application's resource demand. When the application is lightly loaded, the virtual instances of the application are consolidated to lesser number of servers and idle servers are switched to sleep mode to reduce energy consumption. Conversely, when the application's workload is high, some of the VMs from a heavily loaded server are shifted to other servers with sufficient computing power at disposal. The pseudo code for *plan phase* is as shown in Algorithm 6.1.

Execute: The decisions about the mode of operation, which are taken in *Plan* phase of SOS, are applied in this phase. The mode of operation can be switched by changing the control variable used in the resource allocation policy. The second action that needs to be carried out in this phase is migration of VMs, as per the reallocation map that have been evaluated in the *Plan* phase.
The reallocation process includes selection of a server on which the VM can be deployed. The next subsection discusses criterion for selecting a suitable server for VM reallocation.

6.1.1 Criterion for Server Selection

There should be some method to determine the quality of reallocation solution. The reallocation solutions are required to be compared to find the best server. The fitness function is handy in such situations. The fitness function value determines the quality of the reallocation. For the purpose of completeness the fitness function discussed in Chapter 5 is described briefly over here.

Algorithm 6.1 Pseudo code for *Plan* Phase

Output: VMs-Resources map (M_{VR})

Fetch utilization history of application i from *performance database*.

Predict resource utilization of the application.

if $u_i < T_l$ **then**

set $\theta_j = 0$

find underloaded servers where the instances of application i are running.

Consolidate the underloaded servers.

Switch idle servers to sleep mode.

end if

if $u_i > T_u$ **then**

set $\theta_j = 1$

Find overloaded servers where the instances of loaded application i are running.

Identify the VMs for migration from overloaded servers.

Reallocate the migratable VMS to other suitable servers.

end if

if $u_i > T_l$ and $u_i > T_u$ **then**

set $\theta_j = 0.5$

end if

The suitability of resource r for application j is determined from fitness function $f_{j,r}$ given by equation (6.1).

$$f_{j,r} = \frac{\left[\frac{\Re_r^a}{\Re_j^d}\right]^{\theta_j}}{\left[\gamma \triangle E_{j,r} + (1-\gamma)\kappa_r \underbrace{(1 - \sum_{i \in S, i \neq j} \Re_{i,r})}_{y}\right]^{1-\theta_j}}, \qquad \forall r, \tag{6.1}$$

where $\Re_r^a$ is available processing power of resource r, $\Re_j^d$ is processing demand of j^{th} application. $\triangle E_{j,r}$ is the energy contribution of application j on resource r, κ_r is energy affinity, $\Re_{i,r}$ is the fraction of processing power allocated to application i on resource r, and $0 \leq \gamma \leq 1$ is a constant. $0 \leq \theta_j \leq 1$ is a trade off between performance

and energy. By changing the value of θ_j, SOS can be operated in one of the three modes, namely: (i) Power aware, (ii) Performance aware, or (iii) Balanced, mode. The fitness function is used to find a suitable server for reallocation. The server selection process is described in the forthcoming subsection.

6.2 Server Selection

It is desirable to discover the most appropriate physical server for the execution of the application. The selection of a physical server is a complex task because of a large number of heterogeneous physical servers and fluctuations in resource demands of the applications. The server selection process for an application execution should have a minimum delay in terms of total time taken to find the most suitable server. In addition, the server selected for the execution of the application should have sufficient free resources to provide desired QoS, and should cause a minimum increase in energy consumption.

The server selection can be performed by either traditional deterministic algorithms [49, 64, 123, 129, 130, 197] or metaheuristic algorithms [70, 100, 101, 110–112, 218]. Deterministic algorithms suffer from local optima entrapment i.e. they got struck in local solutions and consequently fail to find the true global optimal solution. Moreover, resource allocation using deterministic algorithms is NP-hard. So in the recent years metaheuristic algorithms have been employed for searching the best resource for the application [70, 100, 101, 103–112, 218, 219]. Antlion optimization is preferred over others existing metaheuristic algorithms used in [70, 100, 101, 103–112, 218, 219] because it provides very competitive results in terms of improved exploration, local optima avoidance, exploitation, and convergence [201]. The Antlion optimization algorithm starts with initial random solutions that are represented by the positions of the Antlions. The random solutions are then improved iteratively via random walk of the Ants around the Antlions. The position of an Ant at a particular time during random walk also represents a physical server (solution). The solutions generated during a random walk of an Ant are used to improve the random solution. A new solution for an ant is generated using levy distribution based random walk. The process of generating new

solutions for the purpose of improving initial random solutions is discussed in the next subsection.

6.2.1 Generating new Solutions with Random Walk

The Antlion optimization algorithm starts with some random solutions, called candidate solutions, of the problem at hand, and then improve the candidate solutions iteratively. We get motivation from the way a metaheuristic algorithm operate to find the optimal solution of the problem. When an Antlion optimization algorithm gets trapped in local solution, stochastic operators make random changes in the solution and eventually help in escaping from local optimal solution. The initial random solutions are represented by the Antlions' positions in the search space. These initial solutions are improved by random walks of Ants around the Antlions. The random walk of an Ant is modeled with Levy Flight. The position of an Ant in an iteration represents a solution. The position of Ant at $t+1$ iteration can be evaluated (using Levy Flight) from its position in the previous iteration as shown in equation (6.2):

$$x_i^{t+1} = x_i^t + \alpha \; L(s, \lambda), \tag{6.2}$$

here, x_i^t and x_i^{t+1} represent the position of i^{th} Ant at time t and $t+1$, respectively. α is the scaling factor for step size s. λ is Levy exponent which is a constant, and $L(s, \lambda)$ is Levi distribution with parameters s and λ.

Figure 6.3 shows random walk of an Ant comprising solutions generated in each iteration. In order to restrict the solutions in the range of the search space of the problem at hand, *min-max* normalization is used. For instance, in this work 200 resources are used, which are assigned a unique identification from 1-200, therefore the solutions as shown in Figure 6.3 are normalized to the range 1-200 as given in Figure 6.4.

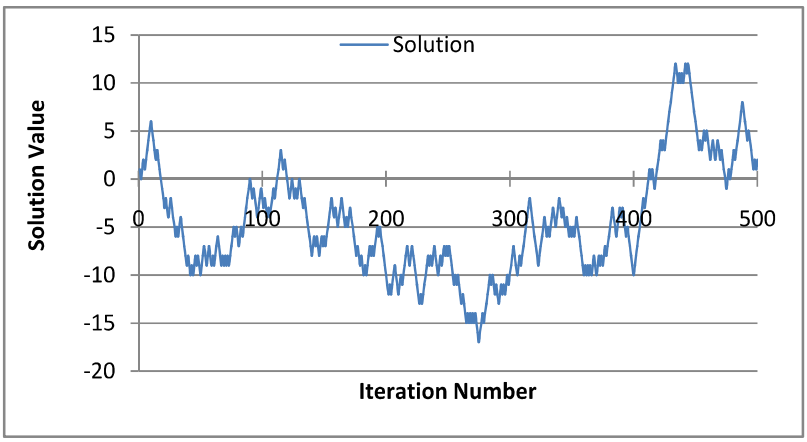


Figure 6.3: Random Walk

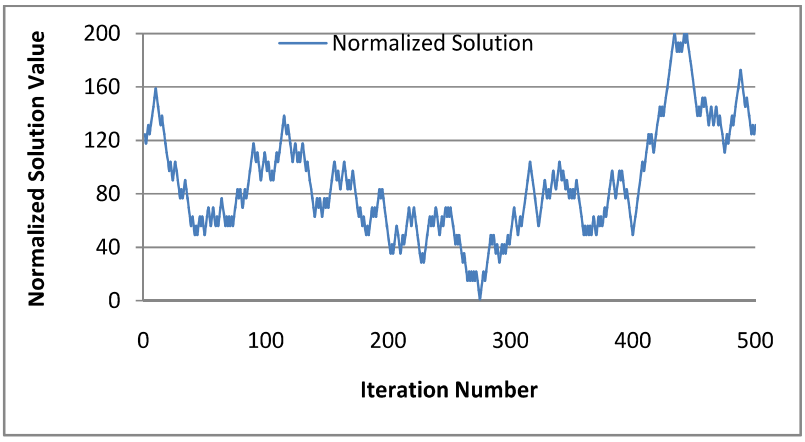


Figure 6.4: Normalized Random Walk

6.3 Resource Allocation using Antlion Optimization

The Antlion optimization algorithm is used for discovering the best resource for a VM. The objective is to find a resource that fulfills not only the resource requirements of the VM but also causes a minimum increase in energy consumption. As discussed earlier, random walk of an Ant generates a new solution in each iteration. Each new solution is compared with the previous solution (represented by an Antlion). When a better solution is found, it replaces the previous solution.

This situation is represented by equation (6.3).

$$V_j^t = W_i^t, \quad \text{if } f(W_i^t) > f(V_j^t), \tag{6.3}$$

where V_j^t is location of j^{th} Antlion at t^{th} iteration, and W_i^t is location of i^{th} Ant at t^{th} iteration.

SOS maintains a record of the best resource (solution). The best solution, called elite, is saved in each iteration. The elite solution has the highest fitness value and effects the random walk of each Ant as shown in equation (6.4).

$$W_i^t = \frac{W_i^t + W_e^t}{2}, \tag{6.4}$$

where W_i^t is a random walk of Ant i around an anlion, and W_e^t is a random walk of elite e at t^{th} iteration.

6.4 Performance Evaluation and Comparative Analysis

The proposed self-optimizing resource allocation system is implemented and tested in CloudSim for evaluation and performance analysis. CloudSim is a framework for modeling and simulation of Cloud computing infrastructure and services [176]. Comparative analysis of SOS is done with state of the art self optimization approaches existing in the literature, i.e. Artificial Bee Colony (ABC) [219], and Self-Optimization of Energy-efficient Cloud resources (SOCCER) [220].

Heterogeneous infrastructure has been created by simulating four different types of servers using *PowerHost* class of CloudSim. The parameters and their corresponding values used in the simulation have been given in Table 6.1. To conduct comparative analysis, ABC, SOCCER, and SOS are tested with Cloudlets (tasks) having diverse CPU utilization requirements. The CPU utilization of tasks is varied as per PlanetLab trace files [221].

The applications are executed in VMs instances that conforms with general purpose

compute: basic tier instances of Microsoft Azure [222]. The aim behind simulating differently dimensioned VMs and servers is to mimic the real Cloud environment such as Amazon, Microsoft Azure, which supports different types of VM instances that are executed on heterogeneous infrastructure. The heterogeneous workload is considered for experimental analysis and results. The various features of cloudlets and resources (allocation parameters) used in the experiments are shown in Table 6.1. The fluctuating workload to the self optimized system has been given in the form of Cloudlets. A Cloudlet, comprising size of the workload, input file size, and output file size, describes the Cloud workload.

6.4.1 Performance Benchmarks

To justify the effectiveness of the self-optimizing system for energy and QoS aware resource allocation, there is a need to define performance evaluation benchmarks. Energy consumption, execution time, number of VMs migrations, number of hot-spots, and number of cold spots have been defined for assessing the performance of the proposed system. Further, a server has been defined as a hot-spot if its resource utilization is above the predefined upper threshold (upper green threshold). Similarly, a server has been assumed a cold-spot if its resource utilization is below a threshold limit (lower green threshold). The lower and upper threshold limits are adopted from the work discussed by Beloglazov *et al.* [94]. Execution time is the total time taken from the time of submission of a workload (Cloudlet) till its completion.

6.4.2 Experimental Results

The proposed self-optimizing system has been tested with 1000 workloads and 200 resources. Statistical accuracy has been ensured through several runs of the simulation experiments. Further, performance analysis for different number of workloads and resources have been carried out through seven test cases as described below.

Test case 1

Comparison of Resource Capacity: The comparison of required resource capacity is

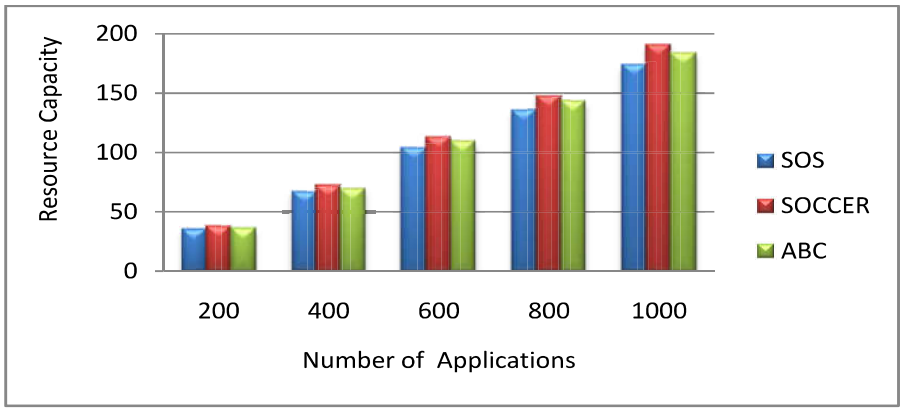


Figure 6.5: Comparison of number of servers required for given workload

evaluated by applying same workload to ABC, SOCCER, and SOS as shown in Figure 6.5. The workload is varied in each simulation run. It is observed that ABC and SOCCER require more number of servers at different workloads. The results show that SOS performs better than the other two approaches when the workload is low. At 200 workloads the servers required for SOS are 13.35% and 9.46% lesser than the number of servers required for ABC and SOCCER, respectively. However, at 1000 workloads, SOS requires 11.27% and 7.66% lesser number of servers than ABC and SOCCER, respectively. Thus, SOS outperforms the other two approaches in terms of resource capacity needed for execution of given workload.

Test case 2

Table 6.1: Simulation Parameters

Number of Applications/VMs	200-1000	Varied in every simulation run
Number of servers	200	the physical machines/servers
Size of worload	10000+(5-30%)MI	task size in millions of instructions(MI)
Input file size of workload	400 + (5-40%) MB	Varied from 5-40% of 400
Output size of workload	400 + (5-50%) MB	Varied from 5-50% of 400
Simulation Span	86400s	Time period for which the simulation is run
T_u	0.85	Upper Green Threshold limit
T_l	0.20	Lower Green Threshold limit

Comparison of Energy Consumption: The comparison of energy consumption of ABC, SOCCER, and SOS for different number of workloads is depicted through Figure 6.6. The outcomes of experiments reveal that SOS is more energy efficient than ABC and SOCCER. It is observed that the energy consumed in kWh at 1000 workloads is 13.48% and 10.32% lesser than ABC and SOCCER. The average energy consumption of SOS

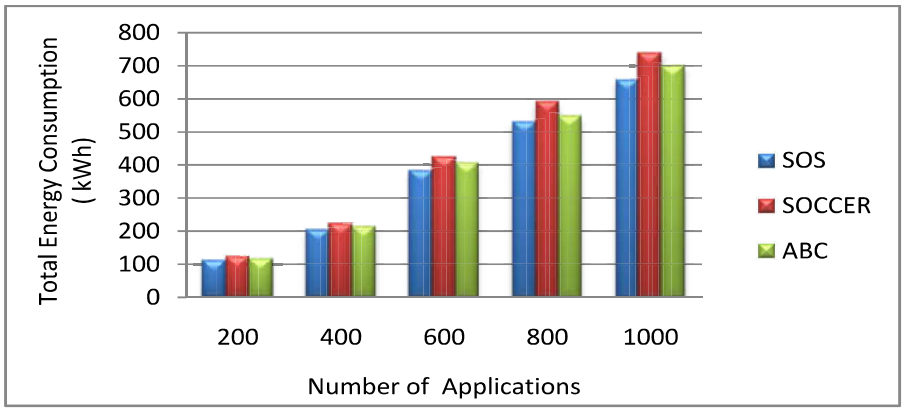


Figure 6.6: Comparison of total energy consumption

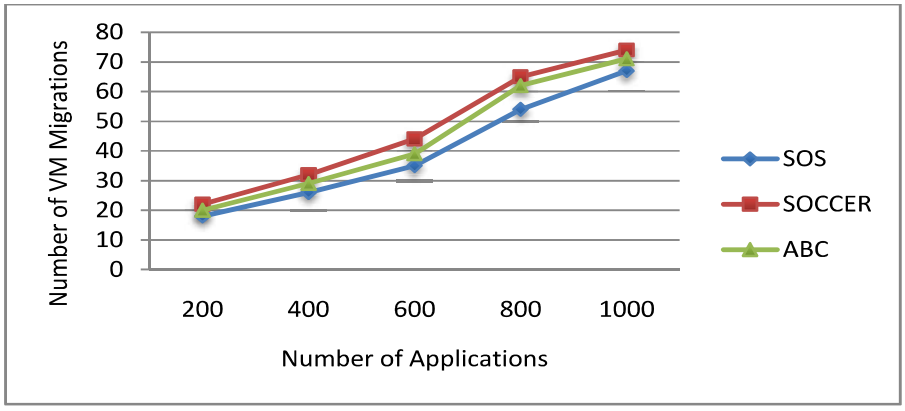


Figure 6.7: Comparison of number of VM migrations

is measured 12.26% and 9.78% lesser than ABC and SOCCER, respectively.

Test case 3

Comparison of Number of Virtual Machine Migrations: Figure 6.7 shows comparison VM migrations performed in ABC, SOCCER and SOS for different workloads. In SOS, VM migration is performed for two purposes: (i) to consolidate servers in order to curb energy consumption. The energy consumption is reduced by switching idle servers to sleep mode, and (ii) to provide better QoS to heavily loaded application by migrating the encapsulating VM to another server having ample resource share at disposal. Whereas, the aim of the authors of ABC and SOCCER is to reduce the energy consumption, so they used VM migration for server consolidation only. It is observed that VM migrations in SOS are 2.4% and 3.6% are more than ABC and SOCCER. The higher number of VM migrations is due to shifting of VMs that are experiencing heavy workload to lighted loaded servers.

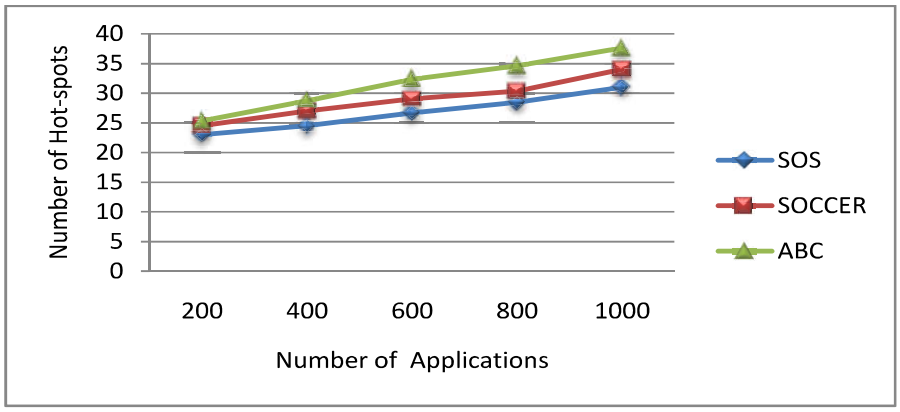


Figure 6.8: Comparison of number of Hot-spots

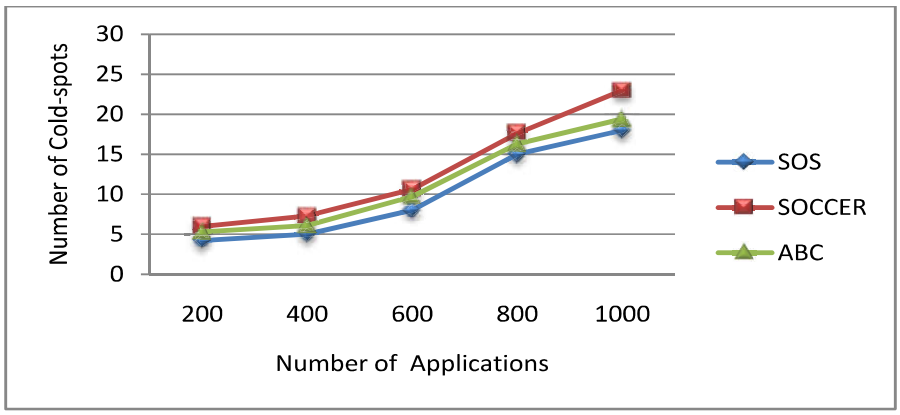


Figure 6.9: Comparison of number of cold spots

Test case 4

Comparison of Number of Hot-Spots: Figure 6.8 shows the comparison of number of hot-spots generated in ABC, SOCCER, and SOS. It is observed that the number of hot-spots increases with the increase in workload. The increase in number of hot-spots in SOS is lesser than that of ABC and SOCCER. The average number of hot-spots in SOS is 3.6% and 2.83% lesser than ABC and SOCCER, respectively. Hot-spots adversely affect the performance and reliability of the servers. Further, creation of hot-spots demands better cooling arrangements.

Test case 5

Comparison of Number of Cold-Spots: The comparative analysis of number of Cold-spots created in ABC, SOCCER and SOS is presented in Figure 6.9. A server is treated as a cold-spot if its resource utilization is below LGT (=20%). The 20% limit is taken because average resource utilization of data center generally remains between 20% and

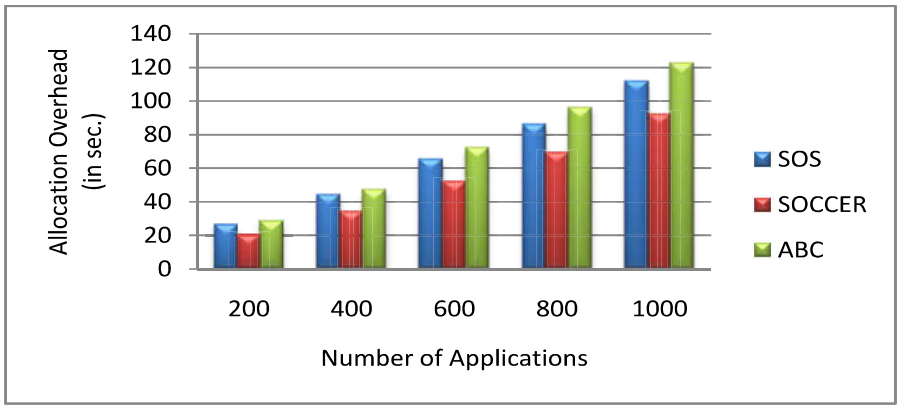


Figure 6.10: Comparison of total mapping time

30% [223]. The number of cold-spots portrays the extent of resource wastage. In SOS 4.6% and 3.76% lesser number of cold-spots than ABC and SOCCER are detected. The fewer cold-spots in SOS shows its ability to allocate the resources efficiently.

Test case 6

Comparison of Overhead: Figure 6.10 depicts a comparison of allocation overhead of ABC, SOCCER and SOS. The time taken by the allocation system to discover a suitable resource for application execution is referred as allocation overhead. The results show that the allocation overhead of SOS is 7.34% and 3.74% lesser than ABC, and SOCCER, respectively.

Test case 7

Comparison of Throughput: Figure 6.11 shows comparative analysis of parameter

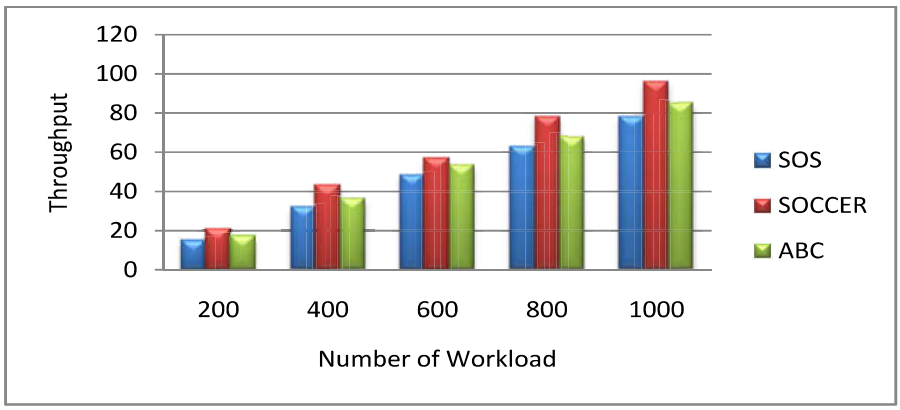


Figure 6.11: Comparison of Throughput

throughput in ABC, SOCCER, and SOS. The simulation span is taken as the time

unit for throughput evaluation. Throughput for SOS is measured 3.4% and 2.85% better than ABC and SOCCER, respectively.

6.5 Summary

In this chapter, a self-optimizing system for energy efficient and QoS aware resource allocation in clouds has been introduced which optimizes the utilization of resources in accordance with fluctuating application's workload. The optimization of resources has been achieved by operating the SOS in one of the supported modes: power aware, performance aware and balanced mode. The proposed system automatically switches between different modes that are inferred from the utilization history of the application. Initial allocation of resources to the applications has been carried out through Antlion optimization. SOS has been implemented in CloudSim and its performance has been compared with existing state-of-the-art self optimization approaches. The results show that SOS outperforms them in terms of energy, performance and QoS.
The next chapter concludes the thesis and also discusses the future directions.

Chapter 7

Conclusion and Future Scope

Energy efficient resource allocation is a challenging task in Cloud Computing due to the potential involvement of extremely large numbers of heterogeneous resources, heterogeneity and fluctuations in the applications' workload. This research work set out to devise energy-efficient resource allocation while respecting QoS requirements of the end users. This thesis efficaciously addresses the resource allocation problem in cloud computing through the proposed green cloud computing framework. The framework is augmented by different energy-efficient and QoS-aware resource allocation policies. Further, a self optimizing resource resource allocation policy is developed that switches its operating mode based on characteristics of the workload. In this chapter, concluding remarks on this research work have been given by describing the advancement made towards the objectives of the research in terms of development of a green cloud framework, and energy and QoS aware resource allocation techniques for deploying heterogeneous applications on cloud resources. The chapter details the outcome of each chapter and later highlights the contributions of the proposed energy-efficient and QoS based allocation policies. Subsequently, it focuses upon the future scope of the study.

7.1 Conclusion

The aim of this research work has been to design and develop energy-efficient resource allocation techniques for cloud computing which has been addressed by the proposed a green cloud framework and energy-efficient and QoS aware resource allocation techniques for Cloud Computing. The objectives set out for this thesis are achieved in phases. The brief summary of each phase is as follows:

(i). In the initial phase, in depth review and analysis of existing literature has been carried out to identify the gaps existing in the area of power management, energy-efficient and QoS-aware resource allocation in Clouds (Chapter 2). The classification of related work on various factors has been done. More specifically, comparative study of various types of resource allocation techniques have been carried out. The existing literature analysis helped to identify research gaps and challenges in the area of energy and QoS aware resource allocation.

(ii). To address the issues of resource allocation, a Green Cloud framework for efficient and robust management of resources is designed and developed as discussed in Chapter 3. The framework has negligible overhead in terms of communication cost. The layered framework is augmented with a resource wastage reduction based allocation technique. The proposed technique has been implemented in simulated as well as real private Cloud environment based on OpenNebula to validate its effectiveness. The proposed allocation approach has shown improvement in energy efficiency upto 18% and 12% in simulated and real private cloud environment, respectively.

(iii). To address the issue of energy wastage due to under-utilization of resources, two level Ant Colony Optimization (ACO) based energy-efficient resource allocation technique has been developed (Chapter 4). The proposed approach minimizes total energy consumption, total execution time, and cost of execution by efficient allocation of resources. At first level, ACO allocates jobs to virtual machines according to their length and associated QoS requirements. Second level ACO decides the placement of virtual machines over physical servers on the basis of geographical distance of data centers and available resource capacity of the physical servers. Further, Dynamic Voltage Frequency Scaling (DVFS) and switching idle machine to sleep mode are employed to reduce energy consumption. The proposed approach achieves upto 10.56% saving in energy consumption through better utilization of resources.

(iv). Providing QoS to the end users while saving energy is the another issue that is addressed in this work. This issue is addressed through a novel Antlion Optimization (AO) based resource allocation approach (Chapter 5). The proposed

approach tunes the values of control variables of an objective function for energy and/or QoS aware resource allocation by adjusting the values of control variables. The approach can be governed to operate in three modes: Energy-aware mode, performance-aware, and balanced mode. In energy-aware mode, the proposed approach strives to use the minimum number of physical resources in order to save energy consumption. In performance mode, the applications are allocated to resources that have the maximum available capacity at disposal. Whereas, in balanced mode, power and performance are given equal weightage while allocating resources to the applications. The best resource is discovered by iterative improvement of objective function value through random walk of Ants around Antlions. Energy savings upto 10.8%, 15.4% and 8.77% are achieved while operating the proposed approach in balanced mode, energy aware mode and performance mode.

(v). The efficient management of large number of resources poses a challenge to the service providers. The challenge is further hardened by the ever changing workload of the applications. This issue is addressed through the proposed self-optimizing resource allocation approach (Chapter 6). The proposed approach is an extension of energy and QoS aware approach presented in Chapter 5. It automatically switches between different operating modes (power, performance, and balanced) inline with the fluctuating workload. The self-optimization resource allocation process has been carried out in four phases: *Monitor*, *Analyze*, *Plan*, and *Execute*. Resource usage history is used for predicting the resource requirements in the near future. The predicted resource requirement is used to decide about the mode of operation for the next time slot. Further, the proposed approach identifies heavily/lightly loaded physical machines to migrate VMs in order to improve energy efficiency/QoS.

7.2 Scope of Future Work

Research is a continuous process. The work presented in this thesis relates to energy and QoS aware resource allocation in clouds. The aim is to allocate optimal resources

in order to improve energy efficiency and QoS. The further research directions in this area are listed below:

(i). The proposed self-optimization system can be extended to support self-healing and self-protecting capability so that it can automatically recover from faults and protect itself against different types of security attacks.

(ii). The proposed work addresses energy consumption and QoS issues of Cloud computing from resource allocation perspective. Other resource management techniques such as resource scheduling can be considered as significant challenges for energy-efficient resource management.

(iii). It would be stimulating to investigate the impact of collocating on the performance and overall energy consumption.

(iv). It would be challenging to provide security and flexibility for mission-critical business applications through efficient resource allocation in Cloud computing.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," http://csrc.nist.gov/publications/nistpubs/800-145/sp800-145.pdf, NIST, 2011.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[3] R. Buyya, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, pp. 599–616, 2009.

[4] J. Voas and J. Zhang, "Cloud computing: New wine or just a new bottle?," *IT Professional*, vol. 11, no. undefined, pp. 15–17, 2009.

[5] M. Bakery and R. Buyyaz, "Cluster computing at a glance," *High Performance Cluster Computing: Architectures and Systems*, vol. 1, pp. 3–47, 1999.

[6] G. V. Mc Evoy and B. Schulze, "Using clouds to address grid limitations," in *Proceedings of the 6th international workshop on Middleware for grid computing*, p. 11, ACM, 2008.

[7] D. Villegas, I. Rodero, L. Fong, N. Bobroff, Y. Liu, M. Parashar, and S. M. Sadjadi, *The Role of Grid Computing Technologies in Cloud Computing*, pp. 183–218. Springer US, 2010.

[8] "Amazon elastic compute cloud." https://aws.amazon.com/ec2/, 2016. Online accessed 2016-03-27.

[9] "Rackspace hosting." https://www.rackspace.com/, 2016. Online accessed 2016-03-12.

[10] "Flexiscale-utility computing on demand." http://www.flexiscale.com/, 2016. Online accessed 2016-01-14.

[11] "Google App Engine." https://appengine.google.com, 2016. Online accessed 2016-04-17.

[12] "Microsoft azure: Cloud computing platform and services." https://azure.microsoft.com/en-in/, 2016. Online accessed 2016-07-05.

[13] "Heroku: Cloud application platform." https://www.heroku.com, 2016. Online accessed 2016-07-17.

[14] "On Demand CRM." https://www.salesforce.com/products/sales-cloud/overview/, 2016. Online accessed 2016-03-07.

[15] "Microsoft Live Mesh." http://connect.microsoft.com/LiveMesh/, 2016. Online accessed 2015-09-23.

[16] "Google Docs." https://docs.google.com/, 2016. Online accessed 20146-03-12.

[17] "Opennebula flexible enterprise cloud made simple." https://opennebula.org/, 2015.

[18] "Openstack open source cloud computing software." https://www.openstack.org/, 2014.

[19] "Oracle Cloud." https://www.oracle.com/cloud/index.html. Online accessed 2014-03-27.

[20] "The open group, service oriented architecture: what is soa?." http://www.opengroup.org/soa/source-book/soa/soa.htm, 2016. Online accessed 2016-11-22.

[21] T. Velte, A. Velte, and R. Elsenpeter, *Cloud Computing, A Practical Approach*. McGraw-Hill, Inc., 1 ed., 2010.

[22] M. Brandner, M. Craes, F. Oellermann, and O. Zimmermann, "Web services-oriented architecture in production in the finance industry," *Informatik-Spektrum*, vol. 27, no. 2, pp. 136–145, 2004.

[23] Y. Wei and M. B. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Computing*, vol. 14, no. 6, pp. 72–75, 2010.

[24] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887.

[25] L. Wang, G. von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010.

[26] E. Ray and E. Schultz, "Virtualization security," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, pp. 42:1–42:5, ACM, 2009.

[27] W. Vogels, "Beyond server consolidation," *Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[28] S. Shumate, "Implications of virtualization for image deployment." http://www.dell.com/downloads/global/power/ps4q04-20040152-Shumate.pdf, 2004. Online accessed 2016-11-22.

[29] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[30] "Folding@home - mysteries of protein folding." https://folding.stanford.edu/home/, 2016.

[31] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing Principles and Paradigms*. Wiley Publishing, 2011.

[32] "Cloud computing versus grid computing-service types, similarities and differences, and things to consider." https://www.ibm.com/developerworks/web/library/wa-cloudgrid/, 2015. Online accessed 2015-03-02.

[33] I. T. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *CoRR*, vol. abs/0901.0131, 2009.

[34] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual workspaces: Achieving quality of service and quality of life in the grid," *Scientific Programming - Dynamic Grids and Worldwide Computing*, vol. 13, no. 4, pp. 265–275, 2005.

[35] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin-Cummings Publishing Co., Inc., 1994.

[36] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European conference on Computer systems*, pp. 265–278, ACM, 2010.

[37] J. Ekanayake and G. Fox, *High Performance Parallel Computing with Clouds and Cloud Technologies*, pp. 20–38. Springer Berlin Heidelberg, 2010.

[38] W. Lu, J. Jackson, and R. Barga, "Azureblast: A case study of developing science applications on the cloud," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pp. 413–420, ACM, 2010.

[39] 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*, vol. 25, no. 6, pp. 599 – 616, 2009.

[40] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

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

[42] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[43] M. A. Vouk, "Cloud computing–issues, research and implementations," in *ITI 2008 - 30th International Conference on Information Technology Interfaces*, pp. 31–40, 2008.

[44] L. Wang, M. Kunze, J. Tao, and G. von Laszewski, "Towards building a cloud for scientific applications," *Advances in Engineering Software*, vol. 42, no. 9, pp. 714 – 722, 2011.

[45] W. Forrest, "How to cut data centre carbon emissions?." http://www.computerweekly.com/feature/How-to-cut-data-centre-carbon-emissions, 2016. Online accessed 2016-07-15.

[46] J. Hamilton, "Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services," pp. 205– 214, Proceedings of CIDR, 2009.

[47] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data center," in *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 826–831, 2010.

[48] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[49] S. Takeda and T. Takemura, "A rank-based vm consolidation method for power saving in datacenters," *IPSJ Transactions on Advanced Computing Systems*, vol. 3, pp. 138–146, 2010.

[50] W. Voorsluys *et al.*, "Cost of virtual machine live migration in clouds: A performance evaluation," *International Conference on Cloud Computing*, vol. 9, pp. 254–265, 2009.

[51] N. Yigitbasi, A. Iosup, and D. Epema, "C-meter: A framework for performance analysis of computing clouds," *Cluster Computing and the Grid*, vol. 9, pp. 472–477, 2009.

[52] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communication ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[53] R. Buyya, A. Beloglazov, and J. H. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *CoRR*, vol. abs/1006.0308, 2010.

[54] H. Tianfield, "Security issues in cloud computing," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1082–1089, 2012.

[55] C. Onwubiko, "Security issues to cloud computing," in *Cloud Computing: Principles, Systems and Applications* (N. Antonopoulos and L. Gillam, eds.), pp. 271–288, Springer London, 2010.

[56] A. Gulati, G. Shanmuganathan, A. Holler, and I. Ahmad, "Cloud-scale resource management: Challenges and techniques," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, pp. 3–3, USENIX Association, 2011.

[57] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424 – 440, 2014.

[58] J. M. Kaplan, W. Forrest, and N. Kindler, "Revolutionizing data center energy efficiency," tech. rep., McKinsy & Company, 2008.

[59] S. V. Vrbsky *et al.*, "Decreasing power consumption with energy efficient data aware strategies," *Future Generation Computer Systems*, vol. 29, pp. 1152–1163, 2013.

[60] J. Koomey, "Growth in data center electricity use 2005 to 2010." http://www.analyticspress.com/datacenters.html, 2014. Online accessed 2015-08-15.

[61] F. Fargo, C. Tunc, Y. Al-Nashif, A. Akoglu, and S. Hariri, "Autonomic workload and resources management of cloud computing services," in *International Conference on Cloud and Autonomic Computing (ICCAC), 2014*, pp. 101–110, 2014.

[62] P. Delforge, "America's data centers consuming and wasting growing amounts of energy." http://www.nrdc.org/energy/data-center-efficiency-assessment.asp. Online accessed 2016-08-15.

[63] M. Webb, *SMART 2020: enabling the low carbon economy in the information age, a report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)*. 2008.

[64] A. Quarati, A. Clematis, A. Galizia, and D. DAgostino, "Hybrid Clouds brokering:Business opportunities, QoS and energy-saving issues," *Simulation Modelling Practice and Theory*, vol. 39, pp. 121–134, 2013.

[65] D. Dahiphale, R. Karve, A. Vasilakos, H. Liu, Z. Yu, A. Chhajer, J. Wang, and C. Wang, "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 101–115, 2014.

[66] J. Hamilton, "The cost of latency," 2015.

[67] A. Kumar, A. Sharma, and R. Kumar, "Servmegh: framework for green cloud," *Concurrency and Computation: Practice and Experience*, 2016.

[68] A. Kumar, R. Kumar, and A. Sharma, "Energy aware resource allocation for clouds using two level ant colony optimization," *Computing and Informatics*, p. in press, 2015.

[69] A. Kumar, R. Kumar, and A. Sharma, "EQUAL: Energy and QoS Aware Resource Allocation Approach for Clouds," *Computing and Informatics*, p. [in press], 2016.

[70] R. Kumar, A. Kumar, and A. Sharma, "A Bio-inspired Approach for Power and Performance Aware Resource Allocation in Clouds," in *4thInternational Conference on Advancements in Engineering and Technology (ICAET-2016)*, p. in press, 2016.

[71] A. Beloglazov, *Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing*. PhD thesis, University of Melbourne, 2013.

[72] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pp. 13–23, ACM, 2007.

[73] M. Blackburn, "Five ways to reduce data center server power consumption," tech. rep., The Green Grid, 2008.

[74] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: Eliminating server idle power," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 1, pp. 205–216, 2009.

[75] L. Minas and B. Ellison, *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*. Intel Press, 2009.

[76] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computer Surveys*, vol. 37, pp. 195–237, 2005.

[77] S. M. S. Devadas, "A survey of optimization techniques targeting low power vlsi circuits," in *32nd Design Automation Conference*, pp. 242–247, 1995.

[78] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," in *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, ISLPED '07, pp. 213–218, ACM, 2007.

[79] "Turbo core technology." http://www.amd.com/en-us/innovations/software-technologies/turbo-core, 2016. Online accessed 2016-03-12.

[80] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova, "Evaluation of the intel® core i7 turbo boost feature," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pp. 188–197, IEEE, 2009.

[81] R. Neugebauer and D. Mcauley, "Energy is just another resource: energy accounting and energy pricing in the nemesis os," in *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, pp. 67–72, 2001.

[82] V. Pallipadi and A. Starikovskiy, "The ondemand governor," in *Proceedings of the Linux Symposium*, vol. 2, pp. 215–230, sn, 2006.

[83] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, "Readings in multimedia computing and networking," ch. Resource Kernels: A Resource-centric Approach to Real-time and Multimedia Systems, pp. 476–490, Morgan Kaufmann Publishers Inc., 2001.

[84] G. Wei, J. Liu, J. Xu, G. Lu, K. Yu, and K. Tian, "The on-going evolutions of power management in xen," tech. rep., Intel Corporation, 2009.

[85] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "Kvm: The linux virtual machine monitor," in *Proceedings of the Linux symposium*, vol. 1, pp. 225–230, 2007.

[86] "Kvm power management." http://www.linux-kvm.org/page/PowerManagement, 2016. Online accessed 2016-07-21.

[87] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Workshop on compilers and operating systems for low power*, vol. 180, pp. 182–195, Barcelona, Spain, 2001.

[88] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," *SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 103–116, 2001.

[89] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *Performance Evaluation Review (PER)*, vol. 33, no. 1, pp. 303–314, 2005.

[90] 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*, HotPower'08, pp. 10–10, USENIX Association, 2008.

[91] 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*, vol. 71, no. 6, pp. 732 – 749, 2011. Special Issue on Cloud Computing.

[92] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pp. 193–204, ACM, 2010.

[93] S. Singh and I. Chana, "Q-aware: Quality of service based cloud resource provisioning," *Computers & Electrical Engineering*, vol. 47, pp. 138 – 160, 2015.

[94] 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*, vol. 28, pp. 755–768, 2012.

[95] J. Meier, C. Farre, P. Bansode, S. Barber, and D. Rea, "Fundamentals of web application performance testing," *Book Chapter, Performance Testing Guidance for Web Applications, Microsoft Corporation*, 2007.

[96] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *The Journal of Supercomputing*, vol. 63, no. 2, pp. 561–592, 2013.

[97] E. Kalyvianaki, "Resource provisioning for virtualized server applications," Tech. Rep. UCAM-CL-TR-762, University of Cambridge, Computer Laboratory, 2009.

[98] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center sla violations and power consumption via hybrid resource provisioning," in *2011 International Green Computing Conference and Workshops*, pp. 1–8, 2011.

[99] T. Xie and B. Wilamowski, "Recent advances in power aware design," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 4632–4635, 2011.

[100] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *12th IEEE/ACM International Conference on Grid Computing (GRID)*, pp. 26–33, 2011.

[101] C.-J. Huang, C.-T. Guan, H.-M. Chen, Y.-W. Wang, S.-C. Chang, C.-Y. Li, and C.-H. Weng, "An adaptive resource management scheme in cloud computing," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 382 – 389, 2013.

[102] N. J. Kansal and I. Chana, "Artificial bee colony based energy-aware resource utilization technique for cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1207–1225, 2015.

[103] L. Chimakurthi and S. D. M. Kumar, "Power efficient resource allocation for clouds using ant colony framework," *CoRR*, vol. abs/1102.2608, 2011.

[104] W. Hu, J. Zheng, X. Hua, and Y. Yang, "A Computing Capability Allocation Algorithm For Cloud Computing Environment," *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, vol. 350, pp. 2258–2262, 2013.

[105] X.-f. Liu, Z.-h. Zhan, K.-j. Du, and W.-n. Chen, "Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach," *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*, pp. 41–48, 2014.

[106] G. Portaluri, S. Giordano, D. Kliazovich, and B. Dorronsoro, "A Power Efficient Genetic Algorithm for Resource Allocation in Cloud Computing Data Centers," in *3rd IEEE International Conference on Cloud Networking (CloudNet)*, pp. 58–63, 2014.

[107] A.-P. Xiong and C.-X. Xu, "Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center," *Mathematical Problems in Engineering*, vol. 2014, pp. 1–8, 2014.

[108] D. Kumar and Z. Raza, "A PSO Based VM Resource Scheduling Model for Cloud Computing," *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, pp. 213–219, 2015.

[109] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 0, no. 0, pp. 1–16, 2015.

[110] N. J. Kansal and I. Chana, "Energy-aware virtual machine migration for cloud computing - a firefly optimization approach," *Journal of Grid Computing*, vol. 14, no. 2, pp. 327–345, 2016.

[111] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230 – 1242, 2013.

[112] H. Chen, A. M. K. Cheng, and Y.-W. Kuo, "Assigning real-time tasks to heterogeneous processors by applying ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 1, pp. 132 – 142, 2011.

[113] C. J. Huang *et al.*, "An adaptive resource management scheme in cloud computing," *Engineering Applications of Artificial Intelligence*, vol. 26, pp. 382–389, 2013.

[114] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers and Electrical Engineering*, vol. 40, no. 5, pp. 1621 – 1633, 2014.

[115] R. Nathuji and K. Schwan, "Virtualpower: Coordinated power management in virtualized enterprise systems," *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 265–278, 2007.

[116] S. K. Garg, C. S. Yeo, and R. Buyya, "Green cloud framework for improving carbon efficiency of clouds," *Euro-Par 2011 Parallel Processing*, vol. 6852, pp. 491–502, 2011.

[117] G. G. Casta, A. Nez, P. Llopis, and J. Carretero, "E-mc2: A formal framework for energy modelling in cloud computing," *Simulation Modelling Practice and Theory*, vol. 39, pp. 56 – 75, 2013.

[118] S. Kinger, R. Kumar, and A. Sharma, "Prediction based proactive thermal virtual machine scheduling in green clouds," *The Scientific World Journal*, vol. 13, pp. 92–103, 2013.

[119] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Dirtributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[120] N. Kim, J. Cho, and E. Seo, "Energy-credit scheduler: An energy-aware virtual machine scheduler for cloud systems," *Future Generation Computer Systems*, vol. 32, pp. 128–137, 2014.

[121] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp. 141–147, 2014.

[122] C.-H. Hsu, S.-C. Chen, C.-C. Lee, H.-Y. Chang, K.-C. Lai, K.-C. Li, and C. Rong, "Energy-Aware Task Consolidation Technique for Cloud Computing," in *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 115–121, 2011.

[123] H. Lee, Y.-S. Jeong, and H. Jang, "Performance analysis based resource allocation for green cloud computing," *The Journal of Supercomputing*, pp. 1–14, 2013.

[124] P. Raycroft, R. Jansen, M. Jarus, and P. R. Brenner, "Performance bounded energy efficient virtual machine allocation in the global cloud," *Sustainable Computing: Informatics and Systems*, vol. 4, pp. 1–9, 2014.

[125] T. Chieu, A. Mohindra, A. Karve, and A. Segal, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment," in *IEEE International Conference on e-Business Engineering, 2009. ICEBE '09.*, pp. 281–286, 2009.

[126] J. Xu and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," in *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, pp. 179–188, 2010.

[127] Q. Liang, J. Liang, and F. Zou, "The resource configuration method with lower energy consumption based on prediction in cloud data center," *Journal of Networks*, vol. 9, no. 7, 2014.

[128] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," *Cluster Computing*, pp. 1–15, 2009.

[129] L. Wu, S. Garg, and R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2011*, pp. 195–204, 2011.

[130] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pp. 119–128, 2007.

[131] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, "Virtual machine migration in an over-committed cloud," in *2012 IEEE Network Operations and Management Symposium*, pp. 196–203, 2012.

[132] G. Foster, G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "The right tool for the job: Switching data centre management strategies at runtime," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 151–159, 2013.

[133] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 91–98, 2010.

[134] A. Nathani *et al.*, "Policy based resource allocation in IaaS cloud," *Procedia Engineering*, vol. 28, pp. 94–103, 2012.

[135] S. Son, G. Jung, and S. C. Jun, "An SLA-based Cloud Computing That Facilitates Resource Allocation in the Distributed Data Centers of a Cloud Provider," *The Journal of Supercomputing*, vol. 64, no. 2, pp. 606–637, 2013.

[136] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, pp. 1230–1242, 2013.

[137] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 559–570, 2011.

[138] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[139] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE'11, pp. 1–1, USENIX Association, 2011.

[140] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, vol. 357, pp. 201 – 216, 2016.

[141] K. Chard, K. Bubendorfer, and P. Komisarczuk, "High occupancy resource allocation for grid and cloud systems, a study with drive," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pp. 73–84, ACM, 2010.

[142] K. Chard and K. Bubendorfer, *Using Secure Auctions to Build a Distributed Metascheduler for the Grid*, pp. 569–588. John Wiley & Sons, Inc., 2009.

[143] H. Hu, Z. Li, and H. Hu, "An Anti-cheating Bidding Approach for Resource Allocation in Cloud Computing Environments," *Journal of Computational Information Systems*, vol. 4, no. 60873022, pp. 1641–1654, 2012.

[144] X. Wang, J. Sun, M. Huang, C. Wu, and X. Wang, "A resource auction based allocation mechanism in the cloud computing environment," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pp. 2111–2115, 2012.

[145] D. DING, S.-W. LUO, and L.-H. AI, "Adaptive double auction mechanism for cloud resource allocation," *Journal of communicating*, vol. 33, no. 60873022, pp. 132–140, 2012.

[146] D. Sun, G. Chang, C. Wang, Y. Xiong, and X. Wang, "Efficient nash equilibrium based cloud resource allocation by using a continuous double auction," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, vol. 1, pp. V1–94–V1–99, 2010.

[147] X. Shi, K. Xu, J. Liu, and Y. Wang, "Continuous double auction mechanism and bidding strategies in cloud computing markets," *CoRR*, vol. abs/1307.6066, 2013.

[148] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 495 – 508, 2013.

[149] A. H. Özer and C. Özturan, "An auction based mathematical model and heuristics for resource co-allocation problem in grids and clouds," in *Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009. Fifth International Conference on*, pp. 1–4, 2009.

[150] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, pp. 7–14, 2010.

[151] F. Teng and F. Magoules, "Resource pricing and equilibrium allocation policy in cloud computing," in *10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, pp. –, IEEE Computer Society, 2010.

[152] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805–818, 2016.

[153] A. Archer, C. Papadimitriou, K. Talwar, and v. Tardos, "An approximate truthful mechanism for combinatorial auctions with single parameter agents.," *Internet Mathematics*, vol. 1, no. 2, pp. 129–150, 2003.

[154] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM (JACM)*, vol. 49, no. 5, pp. 577–602, 2002.

[155] "Microsoft, Windows Azure FAQ." https://azure.microsoft.com/en-us/support/faq/, 2016. Online accessed 2014-04-25.

[156] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control," in *International Conference on Autonomic Computing, 2008. ICAC '08.*, pp. 3–12, 2008.

[157] A. Leva, A. V. Papadopoulos, and M. Maggio, "A general control-theoretical methodology for runtime resource allocation in computing systems," in *52nd IEEE Conference on Decision and Control*, pp. 3487–3492, 2013.

[158] T. Patikirikorala, A. Colman, J. Han, and L. Wang, "A multi-model framework to implement self-managing control systems for qos management," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, pp. 218–227, ACM, 2011.

[159] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, pp. 157–168, ACM, 2009.

[160] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu, "Aggressive resource provisioning for ensuring qos in virtualized environments," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 119–131, 2015.

[161] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: Accelerating resource allocation in virtualized environments," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 423–436, 2012.

[162] W. Song, Z. Xiao, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel & Distributed Systems*, vol. 24, no. undefined, pp. 1107–1117, 2013.

[163] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pp. 11–20, ACM, 2010.

[164] B. Viswanathan, A. Verma, and S. Dutta, "Cloudmap: Workload-aware placement in private heterogeneous clouds," in *2012 IEEE Network Operations and Management Symposium*, pp. 9–16, 2012.

[165] F. Koch, M. D. Assuno, C. Cardonha, and M. A. Netto, "Optimising resource costs of cloud computing for education," *Future Generation Computer Systems*, vol. 55, pp. 473 – 479, 2016.

[166] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155 – 162, 2012.

[167] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," in *2010 IEEE International Conference on Services Computing*, pp. 514–521, 2010.

[168] D. S. Hochbaum, ed., *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

[169] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing." https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/, 2011. Online accessed 2013-01-05.

[170] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *2009 IEEE International Conference on Cloud Computing*, pp. 17–24, 2009.

[171] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and C. Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments," in *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pp. 23–32, IEEE Computer Society, 2008.

[172] R. Gupta, S. K. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti, "A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, vol. 2, pp. 39–46, IEEE, 2008.

[173] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Proceedings of the 11th IFIP/IEEE International Conference on Symposium on Integrated Network Management*, IM'09, pp. 327–334, IEEE Press, 2009.

[174] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler, "More than bin packing: Dynamic resource allocation strategies in cloud data centers," *Information Systems*, vol. 52, pp. 83 – 95, 2015. Special Issue on Selected Papers from {SISAP} 2013.

[175] "Amazon ec2 instance types." https://aws.amazon.com/ec2/instance-types/, 2016. Online accessed 2016-07-02.

[176] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. 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 (SPE)*, vol. 41, no. 1, pp. 23–50, 2011.

[177] R. Buyya, "Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pp. 1–, IEEE Computer Society, 2009.

[178] "Haizea." http://haizea.cs.uchicago.edu/documentation.html, 2014. Online accessed 2014-07-15.

[179] S. Gokhale, W. Wong, K. Trivedi, and J. Horgan, "An analytical approach to architecture-based software reliability prediction," in *Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International*, pp. 13–22, 1998.

[180] J. Musa, "Operational profiles in software-reliability engineering," *Software, IEEE*, vol. 10, no. 2, pp. 14–32, 1993.

[181] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons Ltd., 2nd edition ed., 2002.

[182] "Quickstart: Opennebula on ubuntu 14.04 and kvm," 2015.

[183] "Scheduler." http://docs.opennebula.org/4.12/administration/references/schg.html, 2015. Online accessed 2015-03-10.

[184] S. K. Mandal and P. M. Khilar, "Efficient virtual machine placement for on-demand access to infrastructure resourcesin cloud computing," *International Journal of Computer Applications*, vol. 68, no. 12, pp. 6–11, 2013.

[185] "Predefined host attributes." http://docs.opennebula.org/4.12/user/references/template.html, 2015. Online accessed 2015-03-10.

[186] S. Kumari, A. Maheshwari, P. Goyal, and N. Goyal, "Parallel framework for efficient k-means clustering," in *Proceedings of the 8th Annual ACM India Conference*, Compute '15, pp. 63–71, ACM, 2015.

[187] "Data Flow Diagram." https://en.wikipedia.org/wiki/Data_flow_diagram, 2015. Online accessed 2015-01-05.

[188] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Design Automation Conference, 2004. Proceedings. 41st*, pp. 868–873, 2004.

[189] D. P. BERTSEKAS, "Chapter 4 - exact penalty methods and lagrangian methods," in *Constrained Optimization and Lagrange Multiplier Methods* (D. P. BERTSEKAS, ed.), pp. 179 – 301, Academic Press, 1982.

[190] M. Marzolla and R. Mirandola, "Dynamic power management for qos-aware applications," *Sustainable Computing: Informatics and Systems*, vol. 3, no. 4, pp. 231 – 248, 2013.

[191] T. Gurout, T. Monteil, G. D. Costa, R. N. Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with {DVFS}," *Simulation Modelling Practice and Theory*, vol. 39, pp. 76 – 91, 2013.

[192] P. Wang, Y. Qi, and X. Liu, "Power-aware optimization for heterogeneous multi-tier clusters," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 2005 – 2015, 2014.

[193] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Bradford Company, 2004.

[194] P.-Y. Yin and J.-Y. Wang, "Ant colony optimization for the nonlinear resource allocation problem," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1438 – 1453, 2006.

[195] V. X. Tran, H. Tsuji, and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for Web services ," *Simulation Modelling Practice and Theory*, vol. 17, no. 8, pp. 1378 – 1398, 2009.

[196] M. Yue, "A simple proof of the inequality ffd (l) 11/9 opt (l) + 1, l for the ffd bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, no. 4, pp. 321–331, 1991.

[197] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*, pp. 71–75, 2011.

[198] L. Singh and S. Singh, *A Genetic Algorithm for Scheduling Workflow Applications in Unreliable Cloud Environment*, pp. 139–150. Springer Berlin Heidelberg, 2014.

[199] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275 – 295, 2015.

[200] P. Mitra and G. K. Venayagamoorthy, "Empirical study of a hybrid algorithm based on clonal selection and small population based pso," in *2008 IEEE Swarm Intelligence Symposium*, pp. 1–7, 2008.

[201] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software*, vol. 83, pp. 80 – 98, 2015.

[202] 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*, vol. 24, no. 13, pp. 1397–1420, 2012.

[203] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *SIGOPS - Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[204] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, pp. 225–230, 2007.

[205] B. Walters, "Vmware virtual platform," *Linux Journal*, vol. 1999, no. 63es, 1999.

[206] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1787–1796, 2014.

[207] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682 – 692, 2013.

[208] R. Abbott and H. Garcia-Molina, "Scheduling real-time transactions," *SIGMOD Rec.*, vol. 17, no. 1, pp. 71–81, 1988.

[209] X.-S. Yang and S. Deb, "Cuckoo search via levy flights," in *World Congress on Nature Biologically Inspired Computing*, pp. 210–214, 2009.

[210] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 446–452, 2010.

[211] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–5, 2010.

[212] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pp. 105–113, 2011.

[213] "General purpose compute: Basic tier." https://azure.microsoft.com/en-in/pricing/details/virtual-machines/, 2016. Online accessed 2016-02-29.

[214] A. Gulati, G. Shanmuganathan, A. Holler, and I. Ahmad, "Cloud-scale resource management: Challenges and techniques," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, pp. 3–3, USENIX Association, 2011.

[215] "Opennebula 4.14 administration guide." http://docs.opennebula.org/pdf/4.14/opennebula_4.1 2016. Online accessed 2014-03-19.

[216] J. L. Hellerstein, F. Zhang, and P. Shahabuddin, "An approach to predictive detection for service management," in *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on*, pp. 309–322, 1999.

[217] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *2007 IEEE 10th International Symposium on Workload Characterization*, pp. 171–180, 2007.

[218] A. Kumar, R. Kumar, and A. Sharma, "Energy aware resource allocation for clouds using two level ant colony optimization," *Computing and Informatics*, p. [in press], 2015.

[219] N. J. Kansal and I. Chana, "Artificial bee colony based energy-aware resource utilization technique for cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1207–1225, 2015.

[220] S. Singh, I. Chana, M. Singh, and R. Buyya, "SOCCER: Self-Optimization of Energy-efficient Cloud Resources," *Cluster Computing*, pp. 1–14, 2016.

[221] K. Park and V. S. Pai, "Comon: A mostly-scalable monitoring system for planetlab," *SIGOPS - Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.

[222] "Virtual Machines Pricing." https://azure.microsoft.com/en-us/pricing/details/virtual-machines/Windows. Online accessed 2014-03-13.

[223] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: Eliminating server idle power," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, pp. 205–216, 2009.