Energy Efficient Resource Allocation for Cloud Computing

DILIP KUMAR

(Roll: 212CS1347)

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Energy Efficient Resource Allocation for Cloud Computing

A thesis submitted in
in partial fulfillment of the requirements
for the degree of

Master of Technology

in

Computer Science and Engineering

by

Dilip Kumar
(Roll: 212CS1347)

under the supervision of

Prof. Bibhudatta Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela - 769 008, India

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Prof. Bibhudatta Sahoo
Assistant Professor

Certificate

This is to certify that the work in the thesis entitled *Energy Efficient Resource Allocation for Cloud Computing* by *Dilip Kumar*, bearing roll number 212CS1347, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Master of Technology* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Bibhudatta Sahoo

Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor *Prof. Bibhudatta Sahoo* for the continuous support of my M.Tech study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my M.Tech study.

Besides my advisor, I extend my thanks to our HOD, *Prof. S. K. Rath*, *Prof. S. K. Jena*, *Prof. B. Majhi*, *Prof. D. P. Mohapatra*,*Prof. A. K. Turuk*, *Prof. P. K. Sa*, *Prof. K. S. Babu*, textitProf. R. Dash, *Prof. P. M. Khilar*, *Prof.(Ms.) Sujata Mohanty* and *Prof.(Mrs.) S. Chinara* for their valuable advices and encouragement during my M. Tech study.

Last but not the least, I would like to dedicate this thesis to my family for constantly supporting me throughout my life, their love, patience and understanding. Words fail me to express my gratitude to my beloved mother, who sacrificed her comfort for my betterment.

Dilip Kumar
Roll: 212CS1347
Department of Computer Science and Engineering

Abstract

Cloud computing infrastructures are designed to support the accessibility and deployment of various service oriented applications by the users. Cloud computing services are made available through the server firms or data centers. These resources are the major source of the power consumption in data centers along with air conditioning and cooling equipment. Moreover the energy consumption in the cloud is proportional to the resource utilization and data centers are almost the worlds highest consumers of electricity. The resource allocation problem in a nature of NP-complete, which requiring the development of heuristic techniques to solve the resource allocation problem in a cloud computing environment. The complexity of the resource allocation problem increases with the size of cloud infrastructure and becomes difficult to solve effectively. The exponential solution space for the resource allocation problem can search using heuristic techniques to obtain a sub-optimal solution at the acceptable time. This thesis presents the resource allocation problem in cloud computing as a linear programming problem, with the objective to minimize energy consumed in computation. This resource allocation problem has been treated using heuristic and meta-heuristic approach. Some heuristic techniques are adopted, implemented, and analyzed under one set of common assumptions considering Expected time to compute (ETC) task model for resource allocation. These heuristic algorithms operate in two phases, selection of task from the task pool, followed by selection of cloud resource. A set of ten greedy heuristics for resource allocation using the greedy paradigm has been used, that operates in two stages. At each stage a particular input is selected through a selection procedure. Then a decision is made regarding the selected input, whether to include it into the partially constructed optimal solution. The selection procedure can be realized using a 2-phase heuristic. In particular, we have used 'FcfsRand', 'FcfsRr','FcfsMin','FcfsMax', 'MinMin', 'MedianMin', 'MaxMin', 'MinMax', 'MedianMax', and 'MaxMax'. The simulation results indicate in the favor of MaxMax. The novel genetic algorithm framework has been proposed for task scheduling to minimize the energy consumption in cloud computing infrastructure. The performance of the proposed GA resource allocation strategy has been compared Random and Round Robin scheduling using in house simulator. The experimental results show that the GA based scheduling model outperforms the existing Rondom and Round Robin scheduling models.

Contents

Certificate ii

Acknowledgment iii

Abstract iv

Contents v

List of Figures viii

Abbreviations x

Symbols xii

1 Introduction 2
- 1.1 Cloud Computing . . . 2
- 1.2 Energy Efficient Cloud Computing . . . 2
- 1.3 Resource Allocation . . . 3
- 1.4 Related Works . . . 5
- 1.5 Motivation . . . 6
- 1.6 Problem Statements . . . 8
- 1.7 Research Contributions . . . 8
- 1.8 Layout of the Thesis . . . 9

2 Energy Efficient Cloud Computing Infrastructure, System Model and Performance Parameter 11
- 2.1 Introduction . . . 11
- 2.2 Cloud Computing System Model . . . 13
  - 2.2.1 Energy Consumption in Cloud . . . 15
- 2.3 Problem Model for Energy Efficient Resource Allocation . . . 17
- 2.4 Conclusions . . . 20

3 Energy Efficient Task Consolidation using Greedy Approach 22
- 3.1 Introduction . . . 22
- 3.2 Heuristic Task Consolidation Algorithms . . . 23
  - 3.2.1 FCFS to Random Utilized (FcfsRand) . . . 25
  - 3.2.2 FCFS to Round-Robin Utilized (FcfsRr) . . . 26

3.2.3 FCFS to Minimum Utilized (FcfsMin) . . . 27
3.2.4 FCFS to Maximum Utilized (FcfsMax) . . . 27
3.2.5 Minimum to Minimum Utilized (MinMin) . . . 28
3.2.6 Median to Minimum Utilized (MedianMin) . . . 29
3.2.7 Maximum to Minimum Utilized (MaxMin) . . . 29
3.2.8 Minimum to Maximum Utilized (MinMax) . . . 29
3.2.9 Median to Maximum Utilized (MedianMax) . . . 29
3.2.10 Maximum to Maximum Utilized (MaxMax) . . . 30
3.3 Experimental Evaluation . . . 34
3.3.1 Simulation Enviourments . . . 34
3.3.2 Observation Scenario-1: Three Different Heuristic Algorithms 34
3.3.2.1 Observation-01: Resource Utilization of 5000 tasks on 20 ,40 and 60 VMs . . . 35
3.3.2.2 Observation-02: Energy consumption of 5000 tasks on 60 VMs . . . 37
3.3.2.3 Observation-03: Energy Saving . . . 37
3.3.3 Observation Scenario-2: Ten Different Heuristic Algorithms . 38
3.3.3.1 Observation-04 . . . 39
3.3.3.2 Conclusion: Observation-04 . . . 43
3.3.3.3 Observation-05 . . . 43
3.3.3.4 Conclusion : Observation-05 . . . 48
3.3.3.5 Observation-06 . . . 48
3.3.3.6 Conclusion: Observation-06 . . . 52
3.3.4 Observation Scenario-2: Percentage of Energy Saving . . . 52
3.4 Conclusions . . . 53

4 Energy Efficient Task Consolidation using Genetic Algorithm 55
4.1 Introduction . . . 55
4.2 Genetic Algorithm Based Task Scheduling . . . 56
4.2.1 A Genetic Algorithm . . . 56
4.2.2 Encoding . . . 57
4.2.3 Fitness Function . . . 58
4.2.4 Initial Population . . . 58
4.2.5 Selection . . . 59
4.2.6 Crossover . . . 59
4.2.7 Mutation . . . 60
4.2.8 Stopping condition . . . 61
4.3 Simulation Results . . . 61
4.4 Conclusions . . . 63

5 Conclusions and Future Works 65
5.1 Conclusions . . . 65
5.2 Future Works . . . 66

A Dissemination of Work **67**

Bibliography **68**

List of Figures

2.1 Cloud Computing Architecture 14
2.2 Benchmark of power consumption at various CPU utilization[26] . . 16
2.3 Example of arrival tasks list 19

3.1 Example of FCFS to Random Utilization Tasks allocation Table for 20 tasks 26
3.2 Example of FCFS to Maximum Utilization Tasks allocation Table for 20 tasks 28
3.3 Example of Maximum required to Maximum Utilized, Tasks allocation Table for 20 tasks on 10 VMs 33
3.4 Example of Maximum to Maximum Resources Utilized, resource allocation Table for 20 tasks on 10 VMs 33
3.5 Utilization Comparison for tasks on 20 VMs 35
3.6 Utilization Comparison for tasks on 40 VMs 36
3.7 Utilization Comparison for tasks on 60 VMs 36
3.8 Energy Consumption for 5000 tasks on 60 VMs 37
3.9 Energy Saving compared to FCFSRandomUtil for 5000 tasks on 60 VMs 37
3.10 Ten different Heuristic for experiment scenario-2 38
3.11 Energy consumption on 16 VMs 39
3.12 Energy Saving on 16 VMs 40
3.13 Energy consumption on 32 VMs 40
3.14 Energy saving on 32 VMs 41
3.15 Energy consumption on 64 VMs 41
3.16 Energy saving on 64 VMs 42
3.17 Energy consumption on 128 VMs 42
3.18 Energy saving on 128 VMs 43
3.20 Energy saving on 16 VMs 44
3.19 Energy consumption on 16 VMs 44
3.21 Energy consumption on 32 VMs 45
3.22 Energy saving on 32 VMs 45
3.23 Energy consumption on 64 VMs 46
3.24 Energy saving on 64 VMs 46
3.25 Energy consumption on 128 VMs 47
3.26 Energy saving on 128 VMs 47
3.27 Energy consumption on 16 VMs 48

3.28 Energy saving on 16 VMs 49
3.29 Energy consumption on 32 VMs 49
3.30 Energy saving on 32 VMs 50
3.31 Energy consumption on 64 VMs 50
3.32 Energy saving on 64 VMs 51
3.33 Energy consumption on 128 VMs 51
3.34 Energy saving on 128 VMs 52
3.35 Energy saving comparison 53

4.1 Individual Encoding(chromosome) 57
4.2 Example of mid-point crossover(single point) 59
4.3 Example of mutation at random point with 0.2 mutation probability 60
4.4 No of Generation vs fitness level for deciding optimal stopping condition 61
4.5 Task scheduling on 50 VMs in cloud computing infrastructure. . . . 62
4.6 Task scheduling on 100 VMs in cloud computing infrastructure. . . 63

Abbreviations

ACPI	Advanced Configuration and Power Interface
DPM	Dynamic Power Management
DRS	Distributed Resource Scheduler
DVFS	Dynamic Voltage Frequency Scheduling
FFD	First Fit Decreasing
GA	Genetic Algorithm
GAP	Generalized Assignment Problem
GGA	Grouping Genetic Algorithm
HP	Hewlett Packard
LB	Low Boundary
LLC	Limited Look-ahead Control
MGGA	Modified Group Genetic Algorithm
NIC	Network Interface Controller
MIPS	Million Instructions Per Second
PM	Physical Machine
PS	Primary Service
PUE	Power Usage Effectiveness
SAVMP	Simulated Annealing Virtual Machine Placement
SCE	Server Computing Efficiency
STS	Secondary and Tertiary Service
TOPSIS	Technique For Order Performance by Similarity to Ideal Solution
UPS	Uninterruptable Power Supply
VM	Virtual Machine
API	Applications Programming Interface

ASP	Application Service Provider
BI	Business Intelligence
CaaS	Communications as a Service
CPU	Central Processing Unit
CSN	Cloud Service Partner
CSP	Cloud Service Provider
CSU	Cloud Service User
DaaS	Desktop as a Service
DRAM	Dynamic Random-Access Memory
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
IT	Information Technology
IDC	Internet Data Centre
IoT	Internet of Things
ISB	Inter-cloud Service Broker
ISP	Internet Service Provider
NaaS	Network as a Service
OSS	Operations Support System
PaaS	Platform as a Service
PC	Personal Computer
QoS	Quality of Service
SaaS	Software as a Service
SDP	Service Delivery Platform
SDPaaS	SDP as a Service
SLA	Service Level Agreement
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network

Symbols

τ	Time
t_j	j^{th} Task
$t_{(i,j)}$	Expected time to compute of task t_j on Resource R_i
R_i	i^{th} Resource
H_i	i^{th} Host
$U_i(\tau)$	Utilization of i^{th} Resource at time τ
E_i	Energy Consumption by i^{th} Resource
$E(\tau)$	Total Energy Consumption by Cloud Computing System at time τ
E	Total Energy Consumption by Cloud Computing System
p_{max}	The power consumption at peak load.
p_{min}	The power consumption at inactive load.
λ	Task arrival rate

CHAPTER 1

Introduction

Cloud Computing

Energy Efficient Cloud Computing

Resource Allocation

Related Works

Motivation

Problem Statements

Research Contributions

Layout of The Thesis

Chapter 1

Introduction

1.1 Cloud Computing

The cloud computing is based on the concept of *dynamic provisioning*, which is applied to services, computing capability, storage, networking, and information technology infrastructure to meet user requirements. The resources are made available for the users through the Internet and offered on a pay-as-use basis from different Cloud computing vendors.

1.2 Energy Efficient Cloud Computing

Cloud computing infrastructures are designed to support the accessibility and deployment of various services oriented applications by the users[12],[21]. Cloud computing services are made available through the server firms or data centers. To meet the growing demand for computations and large volume of data, the cloud computing environments provides high performance servers and high speed mass storage devices [2]. These resources are the major source of the power consumption in data centers along with air conditioning and cooling equipment [27]. Moreover the energy consumption in the cloud is proportional to the resource utilization and data centers are almost the world's highest consumers of electricity [5]. Due to the

high energy consumption by data centers, it requires efficient technology to design green data center [19]. On the other hand, Cloud data center can reduce the the total energy consumed through task consolidation and server consolidation using the virtualization by workloads can share the same server and unused servers can be switched off. The total computing power of the Cloud data center is the sum of the computing power of the individual physical machine.

Clouds uses virtualization technology in data centers to allocate resources for the services as per need. Clouds gives three levels of access to the customers: SaaS, PaaS , and IaaS. The task originated by the customer can differ greatly from customer to the customer. Entities in the Cloud are autonomous and self-interested; however, they are willing to share their resources and services to achieve their individual and collective goals. In such an open environment, the scheduling decision is a challenge given the decentralized nature of the environment. Each entity has specific requirements and objectives that need to achieve. Server consolidation are allowing the multiple servers running on a single physical server simultaneously to minimize the energy consumed in a data center [38]. Running the multiple servers on a single physical server are realized through virtual machine concept. The task consolidation also know as server/workload consolidation problem [18]. Task consolidation problem addressed in this thesis is to assign n task to a set of r resources in cloud computing environment. This energy efficient resource allocation maintains the utilization of all computing resources and distributes virtual machines in a way that the energy consumption can minimize. The goal of these algorithms is to maintain availability to compute nodes while reducing the total energy consumed by the cloud infrastructure.

1.3 Resource Allocation

Cloud computing resources are managed through the centralized resource manager. The centralized resource manager assigned the tasks to the required VMs. The

resources of cloud data center are available to the users/applications through Virtual Machines (VMs). Virtual Machines are used to meet the resource requirement and run time support for the applications. In particular executing an application for required resource can be made available through two steps: creating an instance of the virtual machine as required by the application (*VMs provisioning*) and scheduling the request to the physical resources otherwise known as *resource provisioning* [27]. The VM here is to describe the operating system concept: a software abstraction with the looks of a computer system's hardware (real machine) [28]. A virtual machine is sufficiently similar to the underlying physical machine running existing software unmodified. The VM technology has become popular in recent years in data centers and cloud computing environments because it has a number of benefits including server consolidation, live migration, and security isolation. Cloud computing is based on the concept of virtualization that encapsulates various services that can meet the user requirement in a cloud computing environment [13]. Virtual machines (VMs) are designed to run on a server to provide a multiple OS environment with the support of various applications. One or more VM(s) can be placed or deployed on a physical machine that meet the requirement for the VM. The task can be scheduled dynamic load balancing between the host in cloud computing environments are achieved using visualization technology.

Task consolidation is a method to maximize utilization of cloud computing resources. Maximizing resource utilization improves the various benefits such as the rationalization of QoS, IT service customization, maintenance, and reliable services, etc. Improvements in physical hosts hardware [35], such as solid state drives, low power CPUs, and energy efficient computer monitors can helped to reduce the energy consumption issue to a certain degree. There have been a considerable amount of research conducted using resource allocation and software approaches, such as scheduling and server consolidation [18] and task consolidation [32].

1.4 Related Works

Galloway et al. [9] has proposed a load balancing techniques for infrastructure as a service (IaaS) for cloud computing. There are many proposed resource utilizing market-based resource management for various computing areas[39, 5] Kusic et al. [17] have modeled the problem of consolidation. The complexity of the model is too high to the optimization of controller even for a small number of nodes, that is not suitable for large-scale real-world problem. Srikantaiah et al. [32] have studied the multi-tiered web-applications problem in virtualized heterogeneous systems in order to minimize energy consumption. To optimization energy consumption, the authors have proposed a heuristic for the multidimensional bin packing problem as an algorithm for workload consolidation. Song et al. [31] have proposed priorities based resource allocation to applications in a multi-application virtualized cluster. The methods requires machine-learning to obtain the optimized results. Verma et al. [36] have modeled the problem of dynamic placement of services in virtually HDC as continuous optimization. The authors have proposed a heuristic approches for the problem. they have used a bin packing problem with variable bin sizes and costs. Cardosa et al. [7] have discuss the problem of energy efficient allocation of VMs in HDC environments. They have used max, min and shares parameters of VMM that represent maximum, minimum, CPU allocated to VMs sharing the same resource. The approach suits only for private Clouds or enterprise environments. Calheiros et al. [6] have studied the problem of mapping VMs on PH for optimizing network communication between VMs, however, the problem has not been to optimize the energy consumption.

A greedy algorithm solving the problem by making the sub-optimal solution at each with the hope of finding a global optimum stage [3]. A greedy does not produce an optimal solution in many problems, but a greedy heuristic may produce a sub-optimal solutions that approximate a global optimal solution in a reasonable time.

Genetic Algorithm (GA) is computational models which are inspired by the evolutionary process in nature. A typical genetic algorithm requires: a generic representation of the solution domain (chromosome) and a fitness function to evaluate the solution domain. In a genetic algorithm, a specific problem is encoded into a chromosome and a population of candidate solutions (called individuals) to an optimization problem is evolved to get the sub-optimal solutions.

Genetic algorithms can be successfully applied to solve job shop scheduling problem [20], and it can also apply in heterogeneous System [22], grid computing [24] and cloud computing [25]. Most of these researches assume that each task has a fixed amount of execution time (in homogeneous system). Braun et al. [4] compare eleven heuristic and meta-heuristic scheduling methods including of a simple GA-based scheduler, Min-Min, Min-Max, Minimum Completion Time algorithms. The experimental study was performed for task scheduler for independent task in distributed heterogeneous computing environment. The task execution time instances have defined using the ETC matrix model proposed by [1]. Zomaya and Teh [40] proposed a dynamic load balancing framework on genetic algorithm that uses a central scheduler approach to handle all load balancing decisions. The effectiveness of central server with load-balancing has been demonstrated for homogeneous distributed computing system. Kang et al. [14] have discussed in maximizing reliability of distributed computing systems with genetic algorithm based task allocation and the task have represented in task graph. This comparison of different heuristic through simulations proves the effectiveness of genetic algorithms on HDCS. Several researchers have used GA for load balancing on cloud computing systems; however the majority of the papers has no specific representation of the genetic algorithm.

1.5 Motivation

Energy efficiency is increasingly important for cloud computing, because the increased usage of cloud computing infrastructure, together with increasing energy

costs. there is a need to reduce the greenhouse gas emissions call for the energy-efficient technologies that decrease the overall energy consumption of computation, storage and communication equipment. Optimum utilization of energy is increasingly important in data centers. The power dissipation of the physical servers is the root cause of power consumption, which leads to the power consumption of the cooling systems. Many efforts have been made to make data centers more energy efficient. One of these is to minimize the total power consumption of these servers in a data center, through task consolidation and virtual machine consolidation. The current research trends on energy efficient resource allocation have identified the following key area for energy-saving techniques in cloud computing infrastructure:

- **Powering down:** Switching off the entire system when not in use or in idle state can be considered a key area of Energy Aware Computing [9].
- **Dynamic voltage and frequency scaling (DVFS):**

  The DVFS technique is used to reduce the heat generated by the chip in two different way. The power saving can be possible by adjusting automatically the operating frequency of the processor with the help of system clock available on board. Which also reduces the heat generated by the chip on operation.
- **Task Consolidation:** Srikantaiah et al. [32] have discused an approach to switch off the idle machine by finding the minimum number of appropriate machine to which the task to be allocated.
- **Resource Scaling:** In this approche the minimum number of resources are assigned to the set of tasks to meet the deadline in such a way that the task will completed before the deadline to minimize the energy.

1.6 Problem Statements

The problem of resource allocation in cloud computing environments has been presented as minimization problem, to minimize the total energy consumed for a set of task. The resource allocation problem in this thesis assumes the centralized cloud is hosted on a data center that is composed of large number of heterogeneous servers. Each of server may be assigned to perform different or similar functions. A cloud computing infrastructure can be model as PM is a set of physical Servers/host PM_1, PM_2, PM_3, . PM_n. The resources of cloud infrastructure can be used by the virtualization technology, which allows one to create several VMs on a physical server/host and therefore, reduces amount of hardware in use and improves the utilization of resources. The computing resource/node in cloud is used through the virtual machine. A computing resources R is a set of virtual machines $VM_1, VM_2, VM_3,\ VM_m$. The tasks to be scheduled in cloud are with three major three attributes such as *task ID*, *arrival time* and *expected time to compute(ETC)*. In particular the problem addressed in our resource deals with the allocation of VM to a set of tasks such that the total energy consumption of cloud computing infrastructure is minimized by maximizing the resource utilization.

1.7 Research Contributions

The research contribution of *Energy Efficient Resource Allocation for Cloud Computing* are summarized as follows:

1. Formulation of mathematical model for energy efficient resource allocation for Cloud Computing.
2. Design and analysis of energy efficient greedy heuristic task consolidation algorithms.
3. Energy efficient task consolidation using genetic algorithm.

1.8 Layout of the Thesis

In this Thesis, the resource allocation problem in a cloud computing environment has been addressed as an optimization problem. This thesis has been organized into five chapters. The ***Chapter 1*** discusses related research outcomes on energy aware scheduling and resource allocation for cloud computing systems. In ***Chapter 2*** we define the model of cloud computing system, task model and energy consumption of the system. Based on this system model, we have defined the problem to minimize the energy in a cloud computing environment. ***Chapter 3*** discusses the heuristic algorithms used in this study with the illustration and Simulation setup. ***Chapter 4*** discusses the Genetic algorithms to find the solution of our problem domain. Finally, conclusions and directions for future research are discussed in ***Chapter 5***.

CHAPTER 2

Energy Efficient Cloud Computing Infrastructure, System Model and Performance Parameter

Introduction

Cloud Computing System Model

Energy Consumption in Cloud

Problem Model for Energy Efficient Resource Allocation

Summary

Chapter 2

Energy Efficient Cloud Computing Infrastructure, System Model and Performance Parameter

2.1 Introduction

Cloud computing infrastructures are designed to support the accessibility and deployment of various service oriented applications by the users [12] [21]. Cloud computing services are made available through the server firms or data centers. To meet the growing demand for computations and large volume of data, the data centers hosts high performance servers and large high speed mass storage devices [2]. These resources are the major source of the power consumption in data center along with air conditioning and cooling equipment [27]. More over the energy consumption in cloud are proportional to the resource utilization and data centers are almost the worlds highest consumers of electricity [5]. Due to the high energy consumption by data centers, it requires efficient technology to design green data center [19]. Cloud data center, on the other hand, can reduce

the energy consumed through server consolidation, whereby different workloads can share the same server using actualization and unused servers can be switched off.

In general the power management in data centre are related structural constraints relating to the *organization of srever racks* and *number of servers per rack* and *position of the server racks on the floor*. The power management of these resources are possible in two different way; *Static power management* and *dynamic power management*. The Static power management deals with fixed power caps to manage aggregate power. Where as the dynamic power management makes the use of informations related to resources consuming power so as to reduce the power requirement dynamically using advanced platform power management technologies [34]. Clouds uses virtualization technology in distributed data centers to allocate resources to customers as they need them. The task originated by the customer can differ greatly from customer to customer. Entities in the Cloud are autonomous and self-interested; however, they are willing to share their resources and services to achieve their individual and collective goals. In such open environment, the scheduling decision is a challenge given the decentralized nature of the environment. Each entity has specific requirements and objectives that need to achieve.

In this thesis, we propose a heuristic algorithm that could be applied to the centralized controller of a local cloud that is power aware. We capture the Cloud scheduling model based on the complete requirement of the environment. We further create a mapping between the Cloud resources and the combinatorial allocation problem and propose an adequate economic-based optimization model based on the characteristic and the structure of the Cloud.

Cloud computing is based on the concept of virtualization that encapsulates various services that can meet the user requirement in a cloud computing environment [13]. Virtual machines(VMs) are designed to run on a server to provide multiple OS environment in the support of various application. Virtual Machines(VMs) are used to meet the resource requirement and run time support for the applications.

In particular executing an application on required resource can be made available through two step: creating instance of virtual machine as required by the application(*VM provisioning*) and scheduling the request to the physical resources other wise known as *resource provisioning* [27].

Server consolidation are allowing the multiple servers running on a single physical server simultaneously to minimize the energy consumed in a data center [38]. Running the multiple servers on a single physical server are realized through virtual machine concept. The task consolidation also know as server/workload consolidation problem [18]. Task consolidation problem addressed in this thesis is to assign n task to a set of r resources in cloud computing environment. This energy efficient load management maintains the utilization of all compute nodes and distributes virtual machines in a way that is power efficient. The goal of this algorithm is to maintain availability to compute nodes while reducing the total power consumed by the cloud[29], [30].

The remainder of this chapeter is organized as follows. In *Section 2* we define the model of cloud computing system, task model and energy consumption of the system. Based on this system model, we have defined the problem model to minimization the energy in cloud computing environment.

2.2 Cloud Computing System Model

The cloud computing system is consists of fully interconnected set of m resources denoted as R These computing resources are the physical machine in cloud data center and refered as host computing system or host in this chapter. These resources are to be allocated on demand to run applications time to time. Figure 2.1 depicts the system model of cloud computing system, that has been referred in this Chapter. We have assumed the centralized cloud is hosted on a data center that is composed of large number of heterogeneous servers. Each of server may be assigned to perform different or similar functions.

The virtualization technologies allow the creation of multiple virtual machine on any of the available physical host. There for a task can be flexibly assigned to any server. Servers can be modeled as a system that consumes energy in idle state to perform maintenance functions and to have all the subsystems ready while it waits for task to arrive. On arrival of task , a VM processes the task and host may spend an additional amount of energy, which depends on the number of resources demanded by the task, it is represented as resource utilization in work load model.

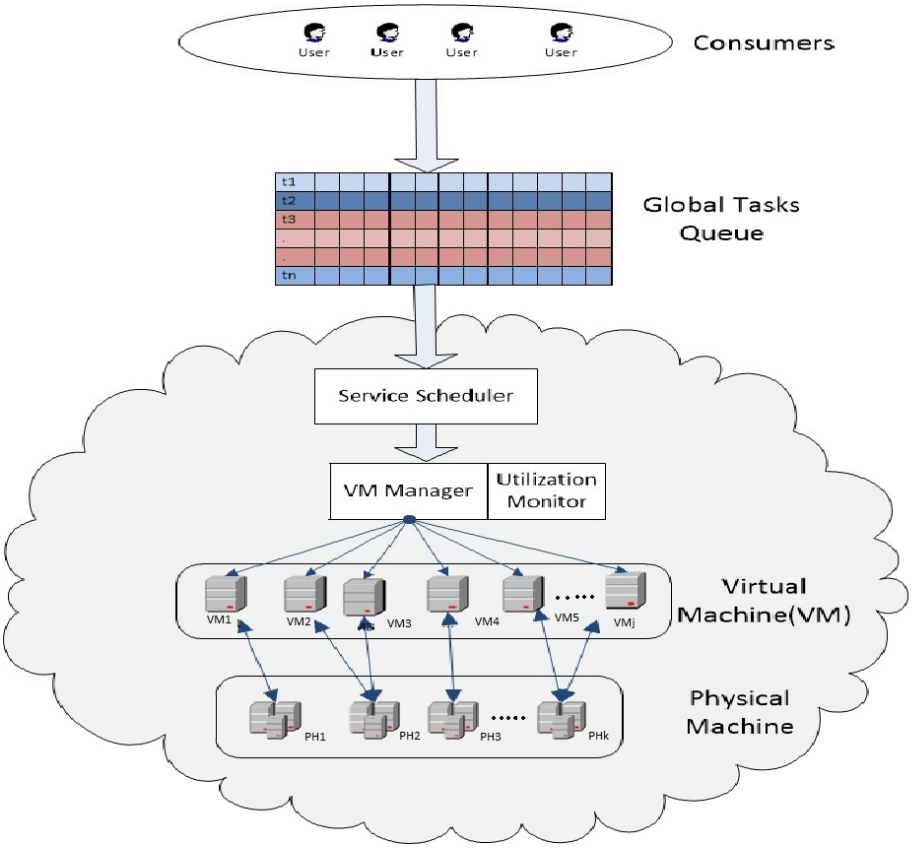


FIGURE 2.1: Cloud Computing Architecture

Although a cloud can span across multiple geographical locations (i.e., distributed), the cloud model in our study is assumed to be confined to a particular physical location.We assume that resources are homogeneous in terms of their computing capability and capacity; this can be justified by using virtualization technologies [18]. It is also assumed that a message can be transmitted from one resource to another while a task is being executed on the recipient resource, which is possible in many systems[18]. The maximum and minimum energy consumption

of the server in cloud computing system are denoted as *pick load state* and *idle state*.

2.2.1 Energy Consumption in Cloud

CPU is the main hardware of a physical machine and its consumed upto 35% of the total energy usages.[8] surveyed a variety of energy models at different levels. So, the computational energy models helps to understand the energy consumption in cloud computing and to develop suitable strategies to improve energy efficiency in cloud computing system.

As formulated in [8], energy consumption is defined as E and characterised for digital static CMOS circuits can be given by

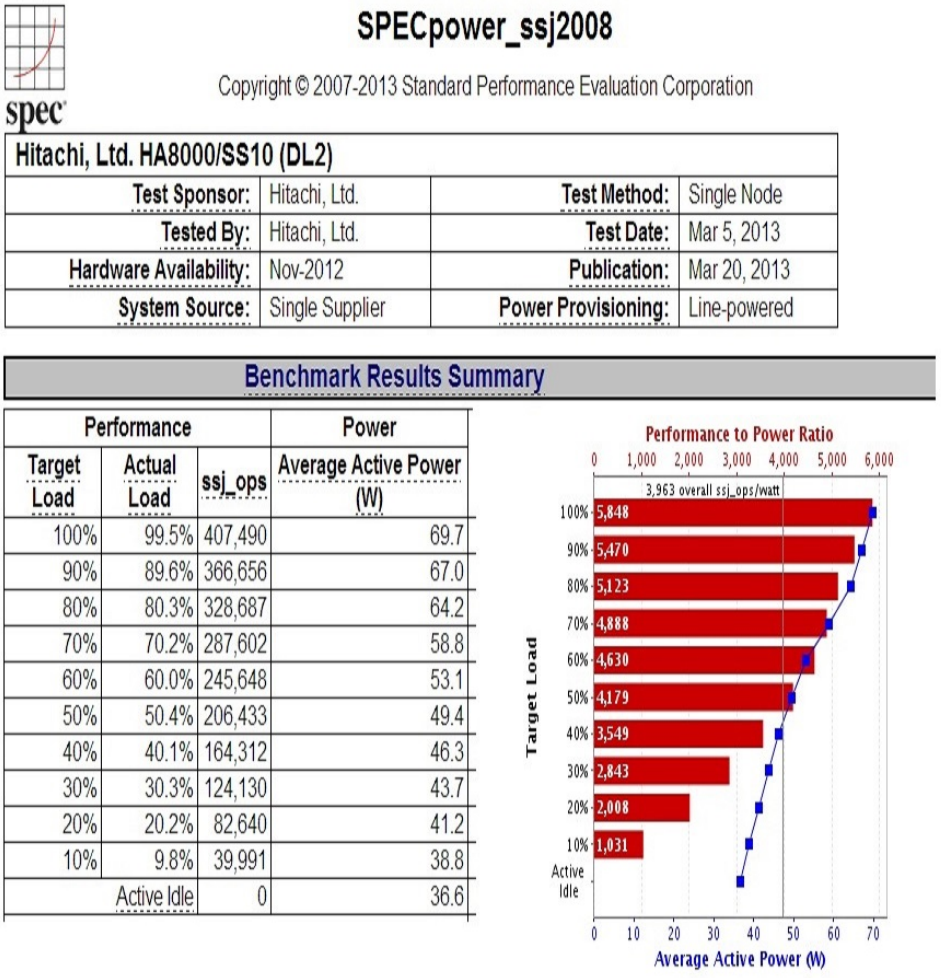
$$E \propto C_{eff} V^2 f_{CLK} \tag{2.1}$$

where C_{eff} is the effective switching capacitance of the operation, V is the supply voltage, and f_{CLK} is the clock frequency. Furthermore, f_{CLK} is relevant to supply voltage as in the equation:

$$f_{CLK} \propto \frac{(V - V_k)^\alpha}{V} \tag{2.2}$$

Equation 2.1 and 2.2 represents the relationships among the energy, voltage and frequency lead to a way of dynamically adjusting voltage and frequency according to the current workloads to conserve energy. However, how much energy can be saved depends largely on the hardware design. Unfortunately many types of server CPU do not have as many levels of voltage and frequency as CPUs for embedded devices, and therefore the power saving acquired by adjusting frequency and voltage vary significantly from one CPU type to another. As CPU is responsible for approximately only one third of the total energy of a typical server, the method of adjusting frequency and voltage only is not enough to solve the power conservation problem.

It is generally believed that the energy consumed by a Physical Machine should be proportional to workloads running on it. However, this is far from true in reality. According to the measurement results by [33] as shown in Figure-2.2, even with nearly zero percentage of CPU utilization, a server can cost up to 50%-60% of the maximum power consumption [37, 26, 11].



SPECpower_ssj2008

Hitachi, Ltd. HA8000/SS10 (DL2)			
Test Sponsor:	Hitachi, Ltd.	Test Method:	Single Node
Tested By:	Hitachi, Ltd.	Test Date:	Mar 5, 2013
Hardware Availability:	Nov-2012	Publication:	Mar 20, 2013
System Source:	Single Supplier	Power Provisioning:	Line-powered

Benchmark Results Summary

Performance			Power
Target Load	Actual Load	ssj_ops	Average Active Power (W)
100%	99.5%	407,490	69.7
90%	89.6%	366,656	67.0
80%	80.3%	328,687	64.2
70%	70.2%	287,602	58.8
60%	60.0%	245,648	53.1
50%	50.4%	206,433	49.4
40%	40.1%	164,312	46.3
30%	30.3%	124,130	43.7
20%	20.2%	82,640	41.2
10%	9.8%	39,991	38.8
	Active Idle	0	36.6

FIGURE 2.2: Benchmark of power consumption at various CPU utilization[26]

This means that it is better to push up the CPU utilization rate to achieve better energy efficiency. However, the system performance may degrade significantly if 100% of CPU or memory utilization is sustained. Instead of 100% resource usage, most servers can handle 70-80% CPU workloads or memory without performance degradation, and high end servers can push the value up to approximately 90% [33]. The energy consumption by host varying with CPU workloads for the whole

machine, the power consumption, which varies with CPU utilization, can be formulated as the equation 2.3 [36, 23, 18]:

$$E(u) = (P_{max} - P_{min}) * \frac{u}{100} + P_{min} \tag{2.3}$$

In the equation, u is the percent value of the processor utilization, E(u) is the Energy consumed by CPU at the utilization u%, and P_{max} and P_{min} are the power consumption at maximum performance in watt and at idle respectively.

2.3 Problem Model for Energy Efficient Resource Allocation

Total Energy E consumed by CPU utilization in time τ by the cloud computing infrastructure by an efficient allocation of resources to the set of tasks. The resource allocation problem on cloud computing are based on following assumptions.

- Virtualization technologies allow the creation of multiple virtual machines on any of the available host.
- Each host may be assigned to perform different or similar services.
- Hosts consumes energy in an idle state to perform maintenance functions and denoted as P_{min}.
- Hosts consumes more energy as per utilization of the CPU by the tasks.
- Hosts consumes maximum energy at the pick level and denoted as P_{max}.
- Hosts put the task in waiting queue, if its CPU utilization is at pick level.

The work load submitted to the cloud is assumed to be in the form of tasks. These tasks are submitted service scheduler. The service schedular allocates the tasks to VMs on different computing hosts. We have assumed the task as the

computational unit to execute on the allocated VM. The task model refered in this chapter are with following assumption.

- A task represents a users computing or service request.
- A task is an independent scheduling entity and its execution cannot be pre-empted.
- The tasks can be executed on any node.
- Arriving task t_j is associated with a *task ID, arrival time, CPU utilization, and expected time to compute* as shown in figure 2.3 for example.
- Tasks arrival rate is Poisson.
- Resource utilization by task is normal distribution between 10% and 100%.
- The resource allocated to a particular task must sufficiently provide the resource usage for that task. If resources are not sufficient, providing the resource usage for a particular task, then task putted in waiting queue.

As shown in figure 2.3 one row of the task arrival list contains the task id, task arrival time, resource utilization by task and estimated execution times for a given task on each machine.

The ETC(t_j,1) indicates the task id, ETC(t_j,2) indicates the task arrival time which is poisson, ETC(t_j,3) indicates the resource utilization by the task t_j and ETC(t_j,4) indiactes the estimated execution times on VM1, and so on.

Task ID	Task Arrival time	Resource utilization(%)	Task Execution Time on VM									
			M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	1	54	12	12	12	12	12	12	12	12	12	12
2	1	62	5	5	5	5	5	5	5	5	5	5
3	1	31	7	7	7	7	7	7	7	7	7	7
4	1	51	12	12	12	12	12	12	12	12	12	12
5	1	97	9	9	9	9	9	9	9	9	9	9
6	2	59	8	8	8	8	8	8	8	8	8	8
7	2	57	11	11	11	11	11	11	11	11	11	11
8	2	31	8	8	8	8	8	8	8	8	8	8
9	2	54	10	10	10	10	10	10	10	10	10	10
10	2	66	10	10	10	10	10	10	10	10	10	10
11	2	71	17	17	17	17	17	17	17	17	17	17
12	3	45	17	17	17	17	17	17	17	17	17	17
13	3	43	13	13	13	13	13	13	13	13	13	13
14	3	99	9	9	9	9	9	9	9	9	9	9
15	3	13	7	7	7	7	7	7	7	7	7	7
16	3	90	12	12	12	12	12	12	12	12	12	12
17	4	93	12	12	12	12	12	12	12	12	12	12
18	4	82	11	11	11	11	11	11	11	11	11	11
19	4	18	6	6	6	6	6	6	6	6	6	6
20	4	33	14	14	14	14	14	14	14	14	14	14

FIGURE 2.3: Example of arrival tasks list

Energy efficient resource allocation for cloud computing can be reprsented as Linear programming problem to minimize the total enegy consumed E, and represented as equation 2.4

$$Minimize\ E\ =\ \sum_{\tau=1}^{\tau}\sum_{i=1}^{m} E_i(\tau) \tag{2.4}$$

Subjected to:

$$E_i(\tau) = (P_{max} - P_{min}) * \frac{U_i(\tau)}{100} + P_{min} \tag{2.5}$$

$$U_i(\tau)\ =\ \sum_{j=1}^{n} u_{(i,j)}\ \leq\ peakload\ at\ time\ \tau,\ \forall\ R_i\ \epsilon\ R\ and\ \forall\ t_j\ \varepsilon\ T \tag{2.6}$$

$$u_{(i,j)} = 0;\ \textit{when the task } j \textit{ is not assigned to node } R_i. \tag{2.7}$$

$$u_{(i,j)} = u_{ij};\ \textit{when the task } j \textit{ is assigned to node } R_i. \tag{2.8}$$

The above equation 2.4 show that the minimization of energy is subjected to the utilization of resources by the task for the time τ.

2.4 Conclusions

In this chapter, we formulated the resource allocation problem as Linear Programming Problem to optimize the energy consumption in cloud computing infrastructure. Heuristics and meta-heuristic technique are preferred by the researchers to address NP-complete problem. The most common heuristic techniques like greedy algorithms, genetic algorithm, PSO, ant colony algorithms, SA, etc are preferred in this researched area. In next chapters, we have used the greedy and genetic algorithms for resource allocation problem.

CHAPTER 3

Energy Efficient Task Consolidation using Greedy Approach

Introduction

Heuristic Task Consolidation Algorithms

Experimental Evaluation

Conclusions

Chapter 3

Energy Efficient Task Consolidation using Greedy Approach

3.1 Introduction

Cloud computing infrastructures are designed to support the accessibility and deployment of various service oriented applications by the users. Cloud computing services are made available through the server firms or data centers. To meet the growing demand for computations and large volume of data, the data centers hosts high performance servers and large high speed mass storage devices. These resources are the major source of the power consumption in data center along with air conditioning and cooling equipment. More over the energy consumption in cloud are proportional to the resource utilization and data centers are almost the world's highest consumers of electricity. The resource allocation problem in cloud computing environment has been shown, in general, to be NP-complete, requiring the development of heuristic techniques. The complexity of resource allocation problem increases with the size of cloud infrastructure and becomes difficult to solve effectively. The exponential solution space for the resource

allocation problem can searched using heuristic techniques to obtained suboptimal solution in the acceptable time. This chapter formulated resource allocation problem in cloud computing as a linear programming problem, with the objective to minimize energy consumed in computation. This chpater uses a set of ten greedy heuristics for resource allocation. All these heuristics from the literature have been selected: adapted, implemented, and analyzed under one set of common assumptions considering ETC task model. These heuristic algorithm operates in two phase, selection of task from the task pool followed by selection of cloud resource. The greedy paradigm provides a framework to design algorithm that work in stages, considering one input at a time. At each stage a particular input is selected through a selection procedure. Then a decision is made regarding the selected input, whether to include it into the partially constructed optimal solution. The selection procedure can be realized using a 2-phase heuristic. In particular we have used 'FcfsRand', 'FcfsRr' ,'FcfsMin' ,'FcfsMax', 'MinMin', 'MedianMin', 'MaxMin', 'MinMax', 'MedianMax', and 'MaxMax'. The simulation results indicate in favor of MaxMax.

3.2 Heuristic Task Consolidation Algorithms

Heuristic and meta-heuristic algorithms are the effective technology for resource allocation problem due to their ability to deliver high quality solutions in reasonable time. The selection procedure can be realized using a 2-phase heuristic. In this section, we present the greedy heuristic algorithms for task allocation in a data center. The general form of task allocation algorithm for the resource utilization of cloud server resources is presented in Algorithm-1.

This algorithm allocate task to the physical resource and maintain the utilization matrix. The Algorithm-1 operates by finding the task which uses maximum resource from the currently available task in task queue.

The function $TaskChoosingPolicy()$ returns the task from the task queue $tempQ$ and the function $ResourceChoosingPolicy()$ returns the resource for the task t_j

for which *maximum threshold value* less then or equal to 100%. If no such fit found it returns null. If resource R_i is found such that utilization is maximum for task t_j and utilization is not exceeding 100%. After allocating task j to resource R_i, the task is removed from the task queue $mainQ$ and temporary queue $tempQ$. If no suitable fit is found then the task j will be removed from temporary queue but not from main queue, this process proceeds to a new iteration. This heuristic algorithm are simple to realize with very little computational cost in comparison to the effort by resource allocation algorithm. The three different heuristic algorithm used in this chapter are described as follows. The algorithm *FCFSMax* have been adapted from heuristic algorithm presented by Lee and Zomaya [18].

Algorithm 1 General Task Allocation Algorithm

Input: Task Matrix

Output: Utilization Matrix

1: Initialize τ
2: Initialize Utilization Matrix, $U^* \leftarrow \phi$.
3: $R^* \leftarrow \phi$.
4: **while** $mainQ \neq \phi$ **do**
5: $tempQ \leftarrow$ All jobs from main queue($mainQ$) where arrival time $\leq \tau$.
6: **while** $tempQ \neq \phi$ **do**
7: $j \leftarrow TaskChoosingPolicy()$
8: $i \leftarrow ResourceChoosingPolicy()$
9: **if** $i \neq Null$ **then**
10: Assign task t_j to R_i
11: Update Utilization Matrix $U_{(\tau,i)}$.
12: Remove task t_j from $mainQ$ and $tempQ$.
13: **else**
14: Remove task t_j from $tempQ$.
15: **end if**
16: **end while**
17: Increment τ.
18: **end while**
19: **return** U.

3.2.1 FCFS to Random Utilized (FcfsRand)

The first heuristic algorithm is known as *FCFSRandomUtil*. This algorithm selects the task in first come first serve (FCFS) basis and the resource is selected in random(using uniform distribution) among the available VMs. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding threshold value 100% including the current task. Iteration continue till all tasks are allocated to VMs.

The example in Figure 3.1 shows time required for the allocation of 20 tasks to 10 VMs.

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	-	[2]	[1]	-	-	[5]	[4]	[3]	-	-
2	[11]	[2]	[1]	[10]	[8]	[5]	[4]	[3]	[6]	-
3	[11,15]	[2]	[1]	[10]	[8]	[5]	[4]	[3]	[6]	-
4	[11,15]	[2]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
5	[11,15]	[2]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
6	[11,15]	-	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
7	[11,15]	[16]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
8	[11,15]	[16]	[1]	[10,20]	[8,12,19]	[5]	[4,13]	[7]	[6]	[14]
9	[11,15]	[16]	[1]	[10,20]	[8,12,19]	[5]	[4,13]	[7]	[6]	[14]
10	[11]	[16]	[1]	[10,20]	[12,19]	-	[4,13]	[7]	[9]	[14]
11	[11]	[16]	[1]	[10,20]	[12,19]	-	[4,13]	[7]	[9]	[14]
12	[11]	[16]	[1]	[20]	[12,19]	-	[4,13]	[7]	[9]	[14]
13	[11]	[16]	-	[20]	[12,19]	[18]	[13]	[7]	[9]	[17]
14	[11]	[16]	-	[20]	[12]	[18]	[13]	[7]	[9]	[17]
15	[11]	[16]	-	[20]	[12]	[18]	[13]	-	[9]	[17]
16	[11]	[16]	-	[20]	[12]	[18]	[13]	-	[9]	[17]
17	[11]	[16]	-	[20]	[12]	[18]	-	-	[9]	[17]
18	[11]	[16]	-	-	[12]	[18]	-	-	[9]	[17]
19	-	-	-	-	[12]	[18]	-	-	[9]	[17]
20	-	-	-	-	[12]	[18]	-	-	-	[17]
21	-	-	-	-	-	[18]	-	-	-	[17]
22	-	-	-	-	-	[18]	-	-	-	[17]
23	-	-	-	-	-	[18]	-	-	-	[17]
24	-	-	-	-	-	-	-	-	-	[17]

FIGURE 3.1: Example of FCFS to Random Utilization Tasks allocation Table for 20 tasks

3.2.2 FCFS to Round-Robin Utilized (FcfsRr)

The *FCFSRRUtil* heuristic algorithm selects the task in first come first serve (FCFS) basis and the resource is selected in round-robin(RR) basis among the available VMs. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding threshold value 100% including the current task. Iteration continue till all tasks are allocated to VMs.

3.2.3 FCFS to Minimum Utilized (FcfsMin)

The task selection process of the *FCFSMinUtil* algorithm also follows FCFS principle. To allocate the selected task, the VM with minimum utilization is selected among the available VMs. The utilization of selected VM is computed by adding the assigned task. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task.

3.2.4 FCFS to Maximum Utilized (FcfsMax)

The task selection process of the *FCFSMaxUtil* algorithm also follows FCFS principle. To allocate the selected task, the VM with maximum utilization is selected among the available VMs. The utilization of selected VM is computed by adding the assigned task. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task. Figure 3.2 the outcome of *MaxUtil* algorithm for 20 tasks to 10 VM.

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	[1,3]	[2]	[4]	[5]	-	-	-	-	-	-
2	[1,3]	[2]	[4]	[5]	[6,8]	[7]	[9]	[10]	[11]	-
3	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10]	[11]	[14]
4	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
5	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
6	[1,3,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
7	[1,3,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
8	[1,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
9	[1,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
10	[1]	[16]	[4,12]	[17]	[18]	[7,13]	[9]	[10,20]	[11]	[14]
11	[1]	[16]	[4,12]	[17]	[18]	[7,13]	[9]	[10,20]	[11]	[14]
12	[1]	[16]	[4,12]	[17]	[18]	[7,13]	-	[20]	[11]	-
13	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
14	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
15	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
16	-	[16]	[12]	[17]	[18]	-	-	[20]	[11]	-
17	-	[16]	[12]	[17]	[18]	-	-	[20]	[11]	-
18	-	-	[12]	[17]	[18]	-	-	-	[11]	-
19	-	-	[12]	[17]	[18]	-	-	-	-	-
20	-	-	-	[17]	[18]	-	-	-	-	-
21	-	-	-	[17]	-	-	-	-	-	-

FIGURE 3.2: Example of FCFS to Maximum Utilization Tasks allocation Table for 20 tasks

3.2.5 Minimum to Minimum Utilized (MinMin)

This algorithm allocate task (which required the minimum resource utilization) to the currently minimum utilizing resources. First the algorithm operated on task queue, which is the resulted on arrival of task till the time of selection. The task is selected from the task queue having minimum resource utilization. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task.

3.2.6 Median to Minimum Utilized (MedianMin)

This algorithm allocate the median task from the sorted task queue to the currently minimum utilizing resources. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task.

3.2.7 Maximum to Minimum Utilized (MaxMin)

This algorithm allocate task (which required the maximum resource utilization) to the currently minimum utilizing resources. The task is selected from the task queue having minimum resource utilization.

3.2.8 Minimum to Maximum Utilized (MinMax)

This algorithm allocate task (which required the minimum resource utilization) to the currently maximum utilizing resources. First the algorithm operated on task queue, which is the resulted on arrival of task till the time of selection. The task is selected from the task queue having minimum resource utilization. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task.

3.2.9 Median to Maximum Utilized (MedianMax)

This algorithm allocate the median task from the sorted task queue to the currently maximum utilizing resources. First the algorithm operated on task queue. The task is assigned to the Virtual Machine R_i, if R_i utilization is not exceeding 100% including the current task.

3.2.10 Maximum to Maximum Utilized (MaxMax)

The pseudo-code for the proposed *MaxMaxUtil* algorithm for the Maximum utilization of cloud server resources is presented in Algorithm-2[16]. This algorithm allocate task (which required the maximum resource utilization) to the currently maximum utilizing resources. First the algorithm operated on task queue, which is the resulted on arrival of task till the time of selection. The task is selected from the task queue having maximum resource utilization. The algorithm 3 $MaximumResourceutilizationTask(temQ)$ return the maximum resource utilizing task from the task queue $tempQ$ and the algorithm 4 $MaximumUtilizingResource(U, \tau, j)$ return the resource which has maximum utilization of resources for task t_j, but less then or equal to maximum threshold value 100% if no such fit found it return 0 value. If resource R_i is found such that utilization is maximum for task t_j and utilization is not exceeding 100%. After allocating task j to resource R_i, the task is removed from the main queue $mainQ$ and temporary queue $tempQ$. If no suitable fit is found then the task j will be removed from temporary queue but not from main queue, the iterative process continue till the successful allocation of all tasks to VMs.

Algorithm 2 MaxMax Algorithm

Input: Task Matrix

Output: Utilization Matrix

1: Initialize τ
2: Initialize Utilization Matrix, $U^* \leftarrow \phi$.
3: $R^* \leftarrow \phi$.
4: **while** $mainQ \neq \phi$ **do**
5: $tempQ \leftarrow$ All jobs from main queue($mainQ$) where arrival time $\leq \tau$.
6: **while** $tempQ \neq \phi$ **do**
7: $i \leftarrow 0$
8: $j \leftarrow MaximumResourceUtilizationTask(tempQ)$
9: $i \leftarrow MaximumUtilizedResource(U, \tau, t_j)$
10: **if** $i \neq Null$ **then**
11: Assign task t_j to R_i
12: $U_{(\tau,i)} \leftarrow U_{(\tau,i)} + utilization(t_j, i)$.
13: Remove task t_j from $mainQ$ and $tempQ$.
14: **else**
15: Remove task t_j from $tempQ$.
16: **end if**
17: **end while**
18: Increment τ.
19: **end while**
20: **return** U.

Algorithm 3 MaximumResourceUtilizationTask Algorithm

Input: Task Queue, TQ

Output: Task id

1: Sort Task queue by utilization in desending order,T
2: retrun(Task id of T(1))

Algorithm 4 MaximumUtilizedResource Algorithm

Input: Utilization Matrix,U; τ; and Task id,j.

Output: Resource id,if fit found otherwise return 0.

1: Temp Utilization Matrix, TempU = ϕ
2: pt=expected time to execute on each machine for task j.
3: **for** $i = 1$ to n **do**
4: **for** $k = 1$ to $pt(i)$ **do**
5: update utilization matrix, tempU(k)= U(τ + k) + utilization(j)
6: **end for**
7: **end for**
8: Remove the resource id, if utilization is more then 100% from tempU.
9: find best fit resource id with maximum utilization, [c,i] = max(sum(tempU))
10: return, i

Allocation list on figure 3.3 is obtained by using algorithm 2 on allocating 20 tasks on 10 VMs in cloud. Figure 3.3 shows the allocation of 20 taks to 10 VMs. The corresponding utilization at a time for 10 VMs is shown in Figure 3.4.

Example of Maximum to Maximum Utilized allocations and utilization are shown in figure 3.3 and figure 3.4 for allocation of 20 tasks to 10VMs.

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	[5]	[2]	[1,3]	[4]	-	-	-	-	-	-
2	[5]	[2]	[1,3]	[4]	[11]	[10,8]	[6]	[7]	[9]	-
3	[5]	[2]	[1,3,15]	[4,12]	[11]	[10,8]	[6]	[7,13]	[9]	[14]
4	[5]	[2]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
5	[5]	[2]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
6	[5]	[17]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
7	[5]	[17]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
8	[5]	[17]	[1,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
9	[5]	[17]	[1,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
10	[16]	[17]	[1]	[4,12]	[11]	[10]	[18]	[7,13]	[9,20]	[14]
11	[16]	[17]	[1]	[4,12]	[11]	[10]	[18]	[7,13]	[9,20]	[14]
12	[16]	[17]	[1]	[4,12]	[11]	-	[18]	[7,13]	[20]	-
13	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
14	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
15	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
16	[16]	[17]	-	[12]	[11]	-	[18]	-	[20]	-
17	[16]	[17]	-	[12]	[11]	-	[18]	-	[20]	-
18	[16]	-	-	[12]	[11]	-	[18]	-	-	-
19	[16]	-	-	[12]	-	-	[18]	-	-	-
20	[16]	-	-	-	-	-	[18]	-	-	-
21	[16]	-	-	-	-	-	-	-	-	-

FIGURE 3.3: Example of Maximum required to Maximum Utilized, Tasks allocation Table for 20 tasks on 10 VMs

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	97	62	85	51	0	0	0	0	0	0
2	97	62	85	51	71	97	59	57	54	0
3	97	62	98	96	71	97	59	100	54	99
4	97	62	98	96	89	97	59	100	87	99
5	97	62	98	96	89	97	59	100	87	99
6	97	93	98	96	89	97	59	100	87	99
7	97	93	98	96	89	97	59	100	87	99
8	97	93	67	96	89	97	59	100	87	99
9	97	93	67	96	89	97	59	100	87	99
10	90	93	54	96	71	66	82	100	87	99
11	90	93	54	96	71	66	82	100	87	99
12	90	93	54	96	71	0	82	100	33	0
13	90	93	0	45	71	0	82	43	33	0
14	90	93	0	45	71	0	82	43	33	0
15	90	93	0	45	71	0	82	43	33	0
16	90	93	0	45	71	0	82	0	33	0
17	90	93	0	45	71	0	82	0	33	0
18	90	0	0	45	71	0	82	0	0	0
19	90	0	0	45	0	0	82	0	0	0
20	90	0	0	0	0	0	82	0	0	0
21	90	0	0	0	0	0	0	0	0	0

FIGURE 3.4: Example of Maximum to Maximum Resources Utilized, resource allocation Table for 20 tasks on 10 VMs

3.3 Experimental Evaluation

The experimental evaluation done through the inhouse discreate event simulation in Matlab2012. We have taken two Scenario to observe the result. We have conducted various experiments on variable number of VMs, and tasks. In first scenario we have used three heuristic algorithms on 5000 tasks to observer the resource utilization, energy consumption and percentage of energy saving. In second scenario, we have observe the results for ten different greedy heuristic algorithms on 500 tasks to see the outcome for energy consumption and energy saving.

3.3.1 Simulation Enviourments

- Matlab 2012 tools has been used for creating the Energy Model, Task Model and implementation of algorithms.
- Power Spec benchmark has been used as power model of server specification.
- All experiments were run on systems with Windows 8 (32 bit) operating system on Intel Core i3 processor.

3.3.2 Observation Scenario-1: Three Different Heuristic Algorithms

In this scenario we have used three heuristic algorithms on 1000 to 5000 tasks to observer the resource utilization, energy consumption and percentage of energy saving. The following observations are:

- Resource utilization of three heuristic algorithms on 20, 40 and 60 VMS, arrival interval 1 and arrival rate 60 with 5000 tasks are Observed. The result of this observation are shown in Figure 3.5, 3.6 and 3.7.

- Energy Consumption of three heuristic algorithms on 60 VMS, arrival interval 1 and arrival rate 60 with 5000 tasks are Observed. The result of this observation are shown in Figure 3.8.

- Energy saving of three heuristic algorithms on 60 VMs, arrival interval 1 and arrival rate 60 with 5000 tasks are Observed. The result of this observation are shown in Figure 3.9.

3.3.2.1 Observation-01: Resource Utilization of 5000 tasks on 20 ,40 and 60 VMs

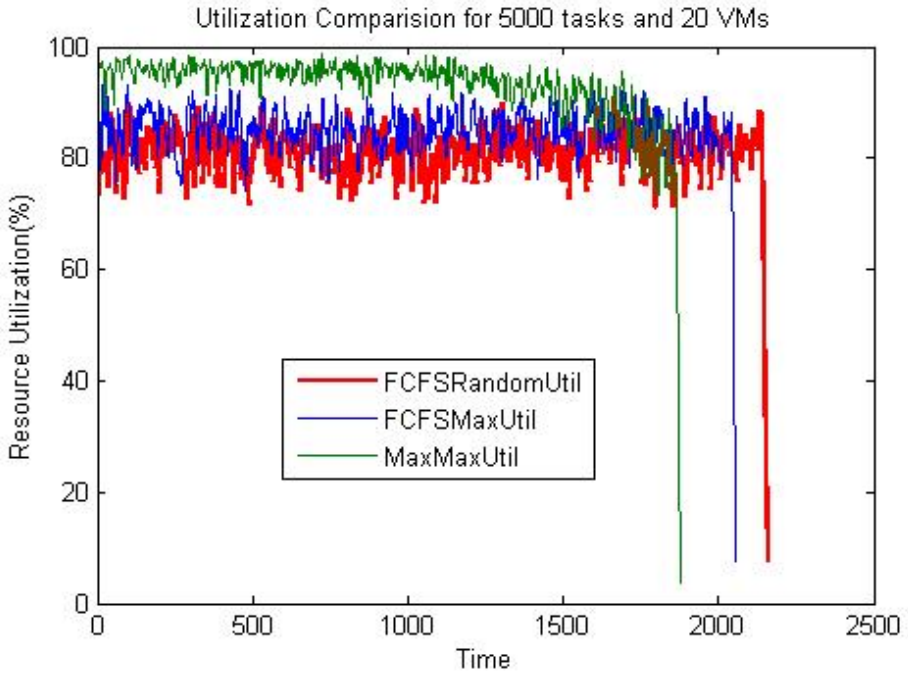


FIGURE 3.5: Utilization Comparison for tasks on 20 VMs

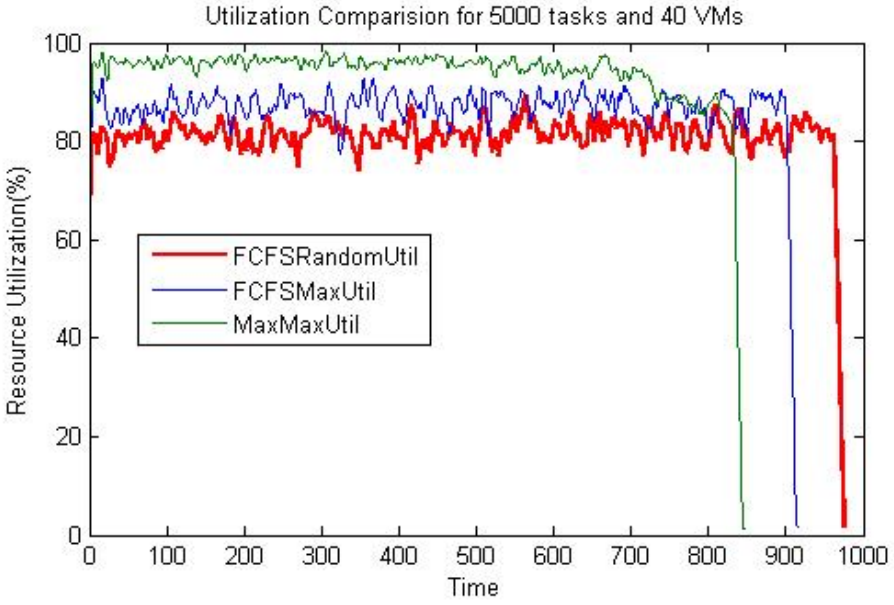


FIGURE 3.6: Utilization Comparison for tasks on 40 VMs

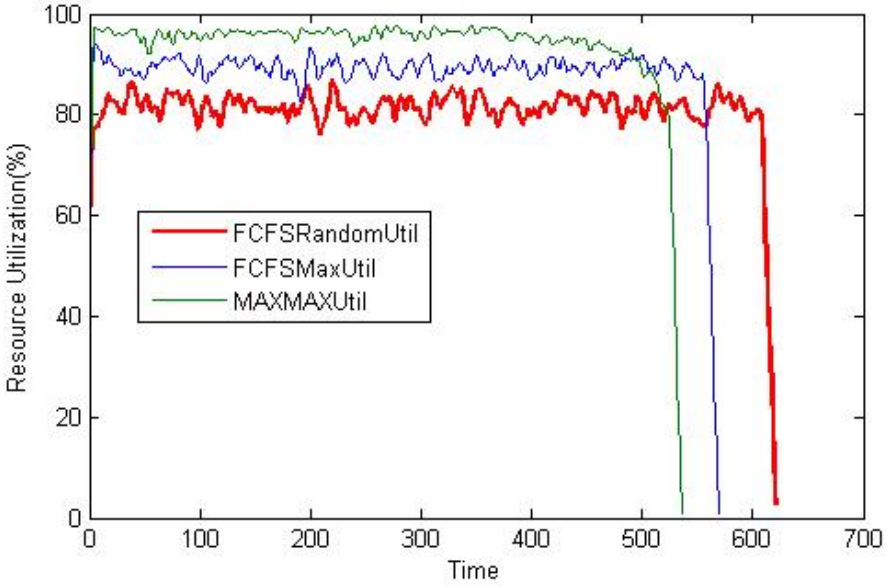


FIGURE 3.7: Utilization Comparison for tasks on 60 VMs

3.3.2.2 Observation-02: Energy consumption of 5000 tasks on 60 VMs

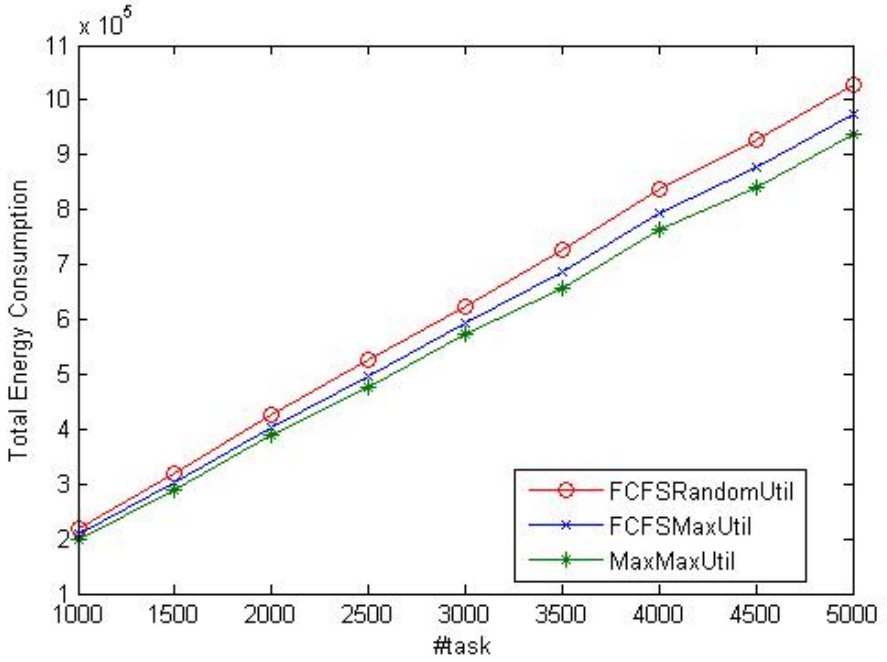


FIGURE 3.8: Energy Consumption for 5000 tasks on 60 VMs

3.3.2.3 Observation-03: Energy Saving

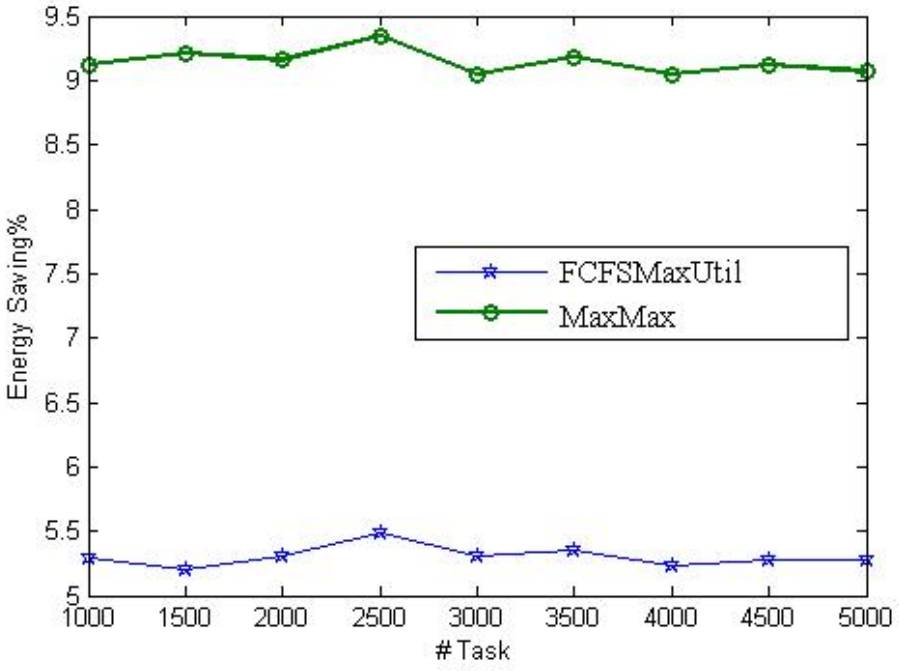


FIGURE 3.9: Energy Saving compared to FCFSRandomUtil for 5000 tasks on 60 VMs

3.3.3 Observation Scenario-2: Ten Different Heuristic Algorithms

In this scenario, we have observe the results for ten different greedy heuristic algorithms on 100 to 1000 tasks to see the outcome for energy consumption and energy saving. We have grouped the 2- stage greedy algorithms based on first stage and second stage in the group of 4. In first group we have taken FCFS as Task selection and Rand, RR, Min, Max utilized resource for resource selection. In second group, the best of first is taken and other three are MinMin, MedianMin and MaxMin. In group three the best of second group is taken and other three are MinMax, MedianMax and MaxMax. The group table used in this scenario is shown in Figure 3.10.

	Phase-1 Task selection	Phase-2 Resource selection	Algorithm Name
Group-1	FCFS	Random	FcfsRand
	FCFS	Round Robin	FcfsRr
	FCFS	Min	FcfsMin
	FCFS	Max	FcfsMax
Group-2	Best of group-1		
	Min	Min	MinMin
	Median	Min	MedianMin
	Max	Min	MaxMin
Group-3	Best of group-2		
	Min	Max	MinMax
	Median	Max	MedianMax
	Max	Max	MaxMax

FIGURE 3.10: Ten different Heuristic for experiment scenario-2

The following experiments have been conducted for ten diffrent heuristic algorithms in group of four.

- Energy consumption with ten different heuristic algorithms on 16, 32, 64, and 128 VMs, arrival interval 1 and arrival rate 60 with 100 to 1000 tasks are observed in a group of 4 heuristic algorithms.

- Energy saving with ten different heuristic algorithms on 16, 32, 64, and 128 VMs, arrival interval 1 and arrival rate 60 with 100 to 1000 tasks are observed in group of 4 heuristic algorithms.

3.3.3.1 Observation-04

In this section we observe the energy consumption and energy saving of group-1 (see table 3.10) heuristic algorithms on 20, 40 and 60 VMs for 100 to 1000 tasks. The results are shown in Figure 3.11 to 3.18.

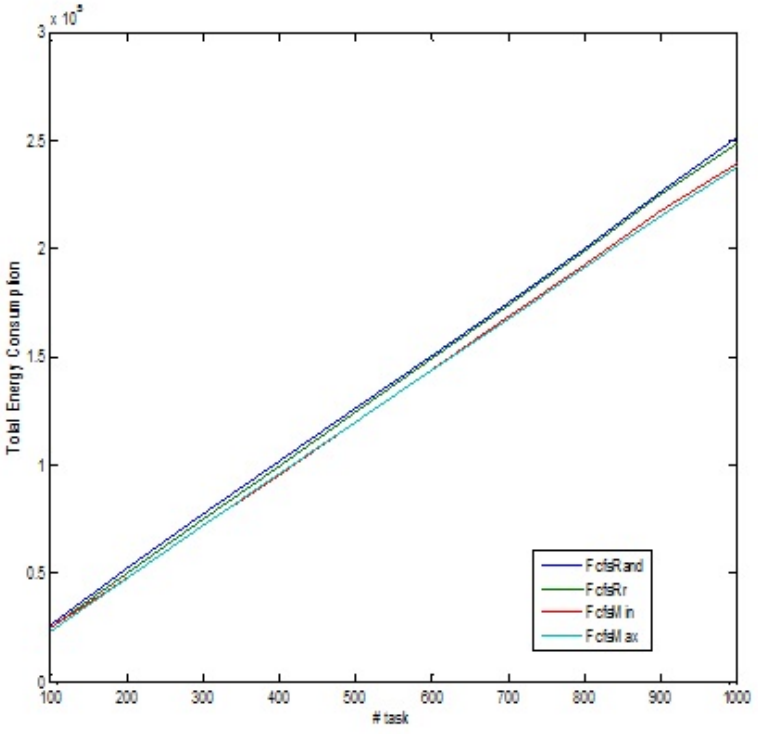


FIGURE 3.11: Energy consumption on 16 VMs

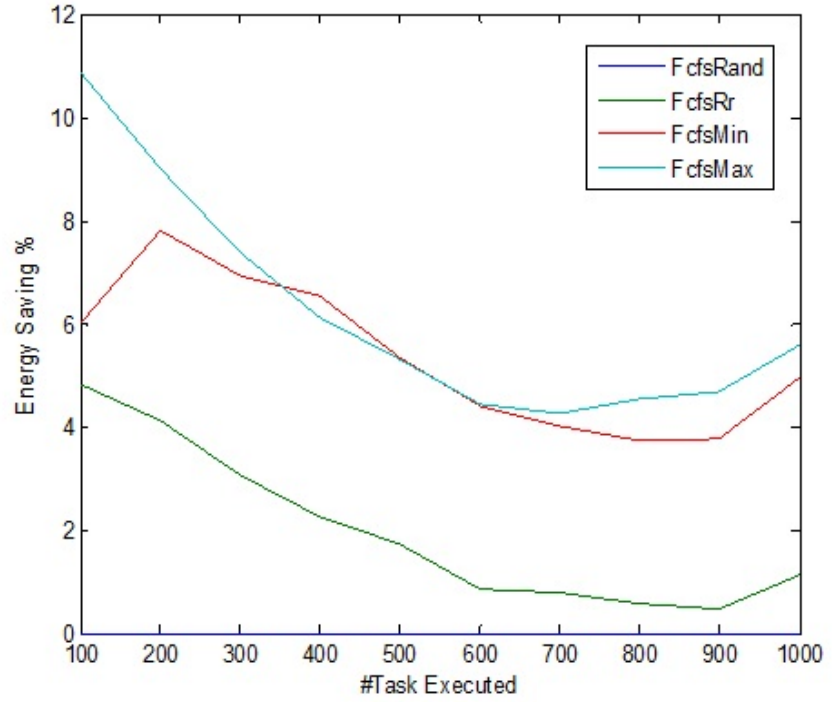


FIGURE 3.12: Energy Saving on 16 VMs

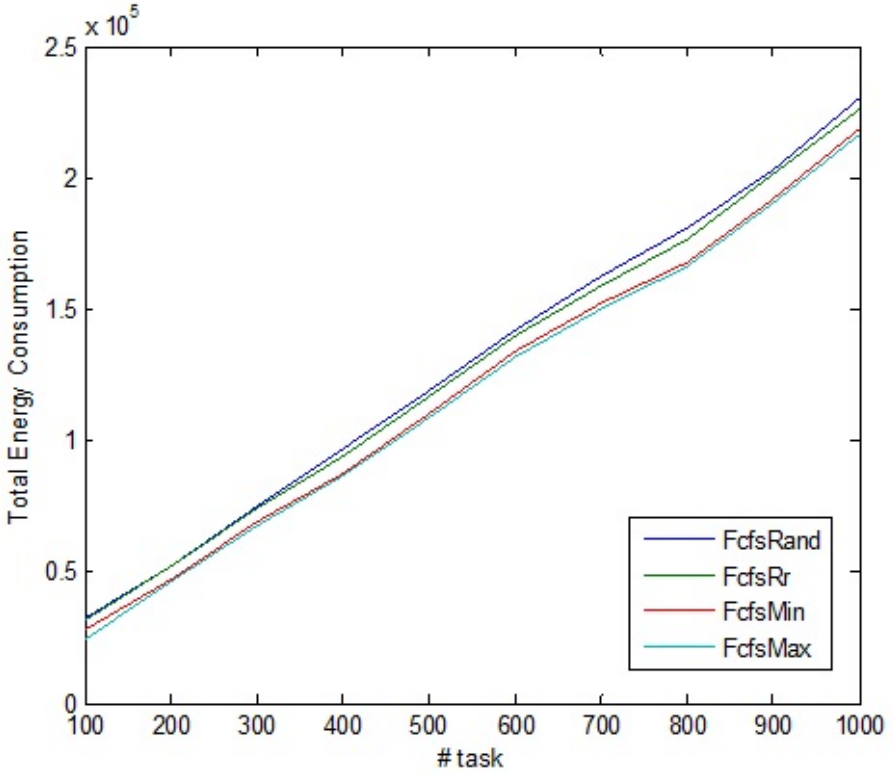


FIGURE 3.13: Energy consumption on 32 VMs

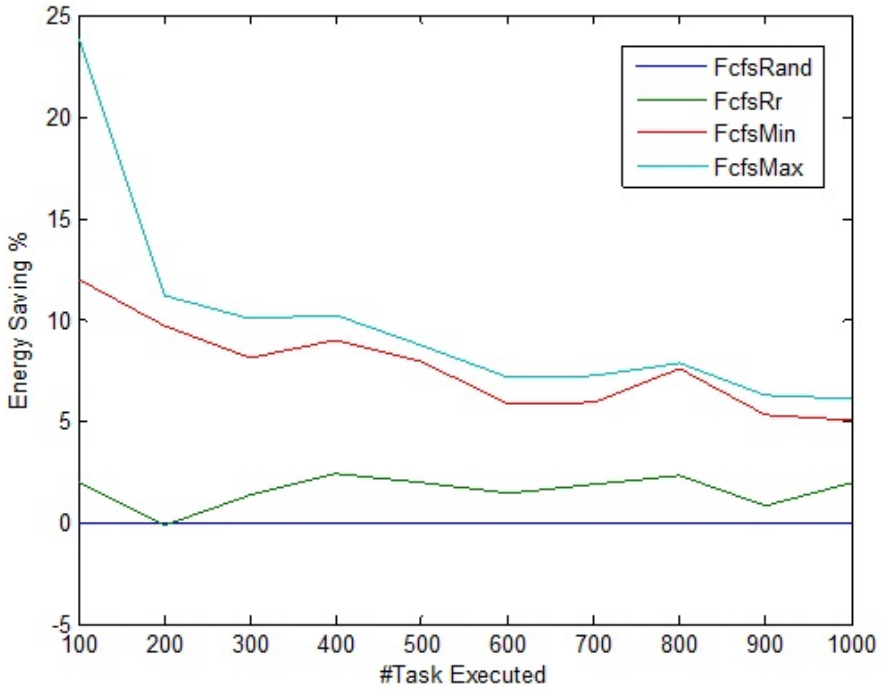


FIGURE 3.14: Energy saving on 32 VMs

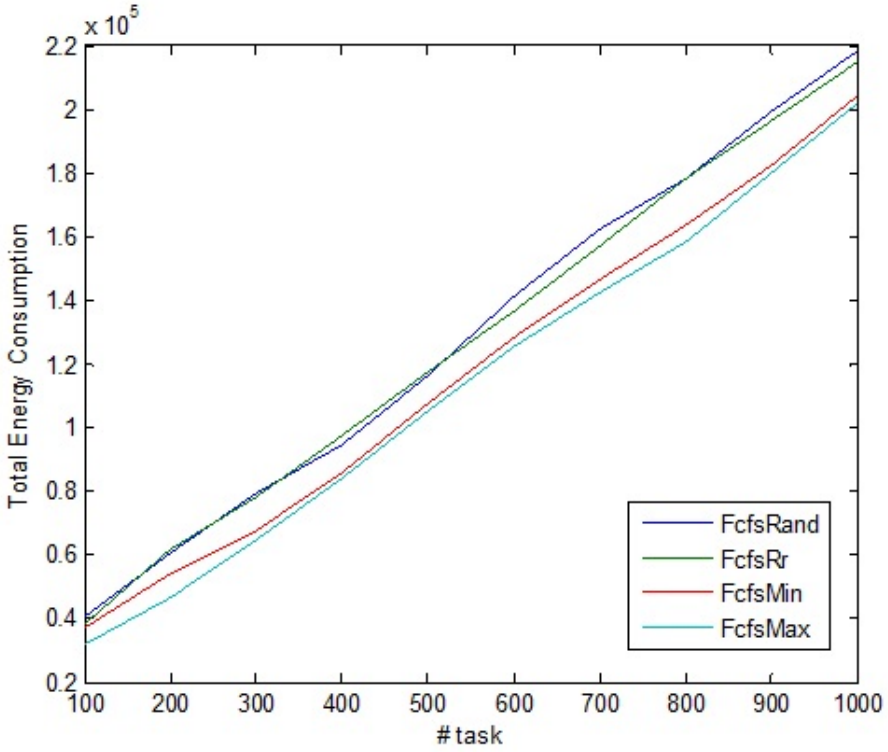


FIGURE 3.15: Energy consumption on 64 VMs

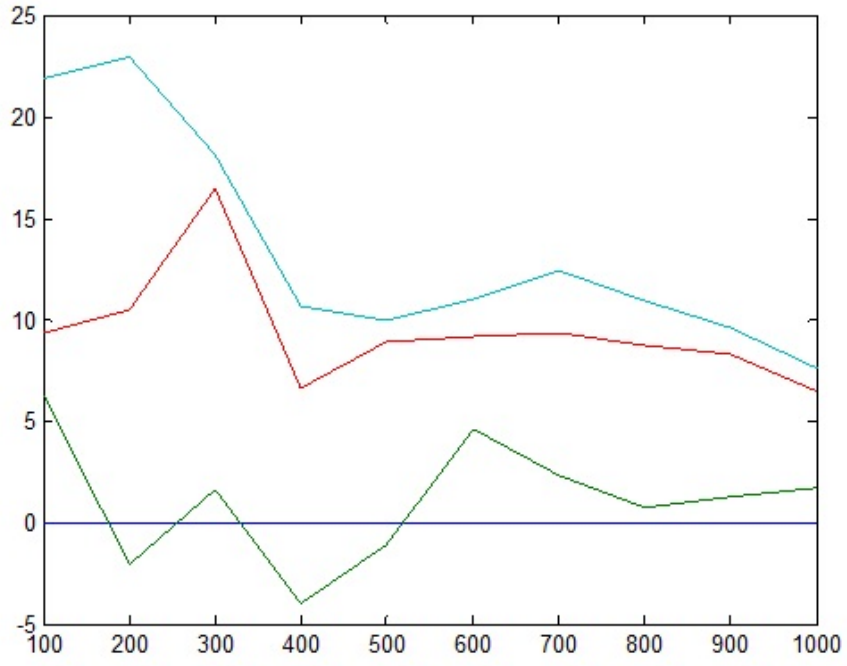


FIGURE 3.16: Energy saving on 64 VMs

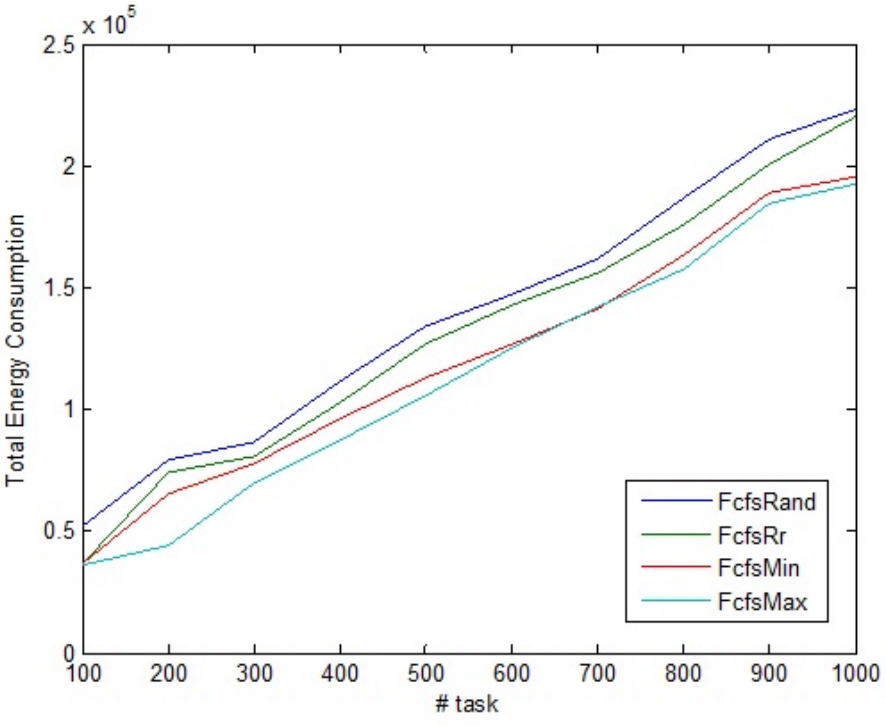


FIGURE 3.17: Energy consumption on 128 VMs

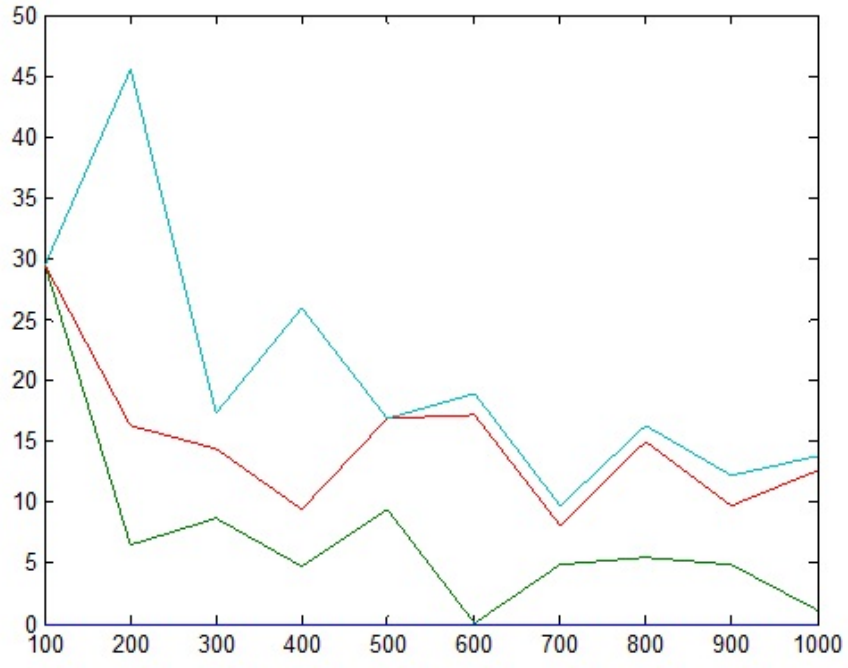


FIGURE 3.18: Energy saving on 128 VMs

3.3.3.2 Conclusion: Observation-04

It is observed that the energy consumption of FCFSMax scheduling is minimum in this group.

3.3.3.3 Observation-05

In this section we observe the energy consumption and energy saving of group-2 (see table 3.10) heuristic algorithms on 20, 40 and 60 VMs for 100 to 1000 tasks. The results are shown in Figure 3.19 to 3.26.

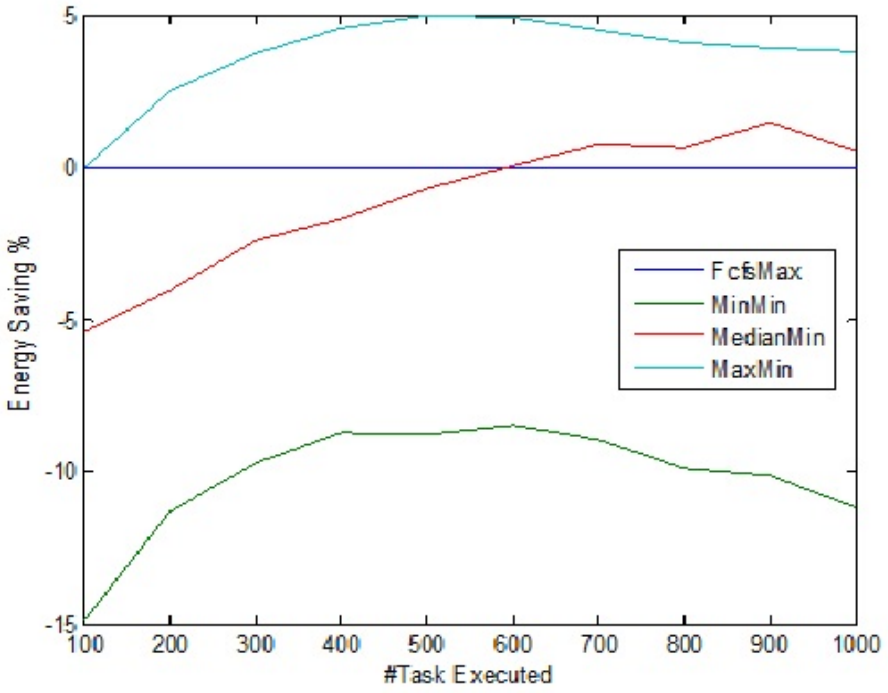


FIGURE 3.20: Energy saving on 16 VMs

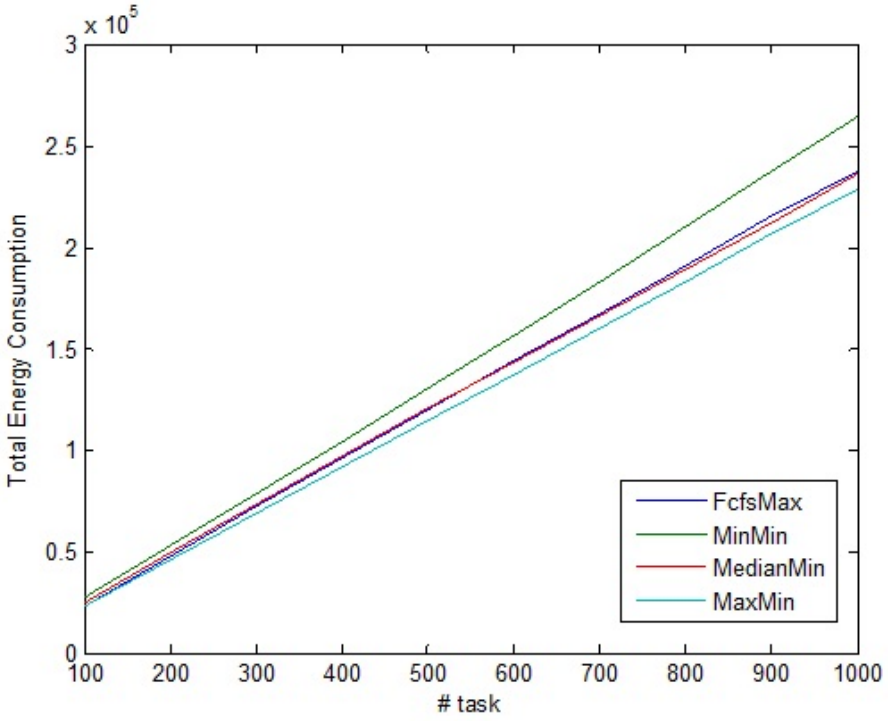


FIGURE 3.19: Energy consumption on 16 VMs

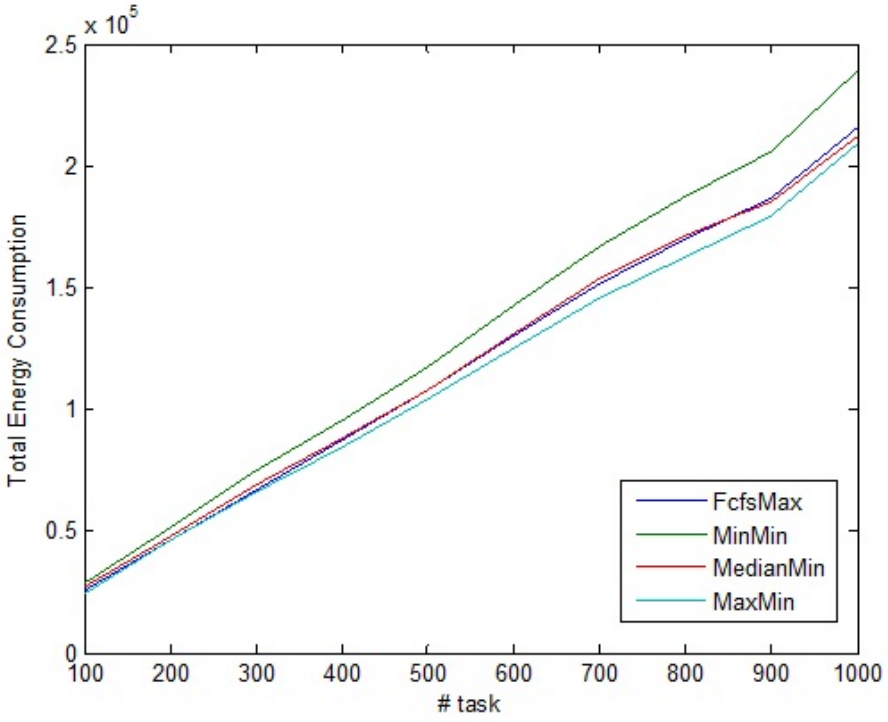


FIGURE 3.21: Energy consumption on 32 VMs

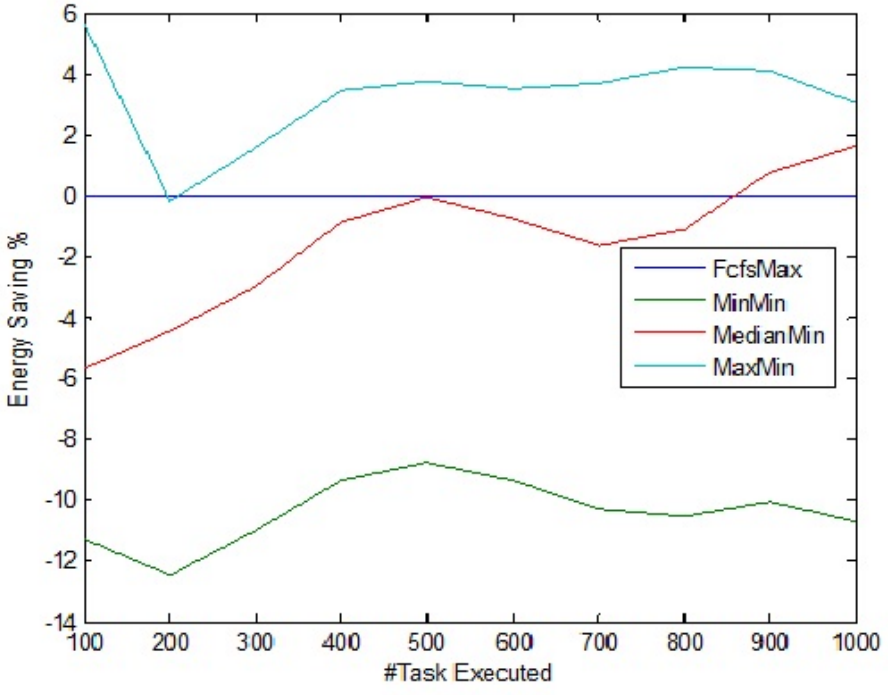


FIGURE 3.22: Energy saving on 32 VMs

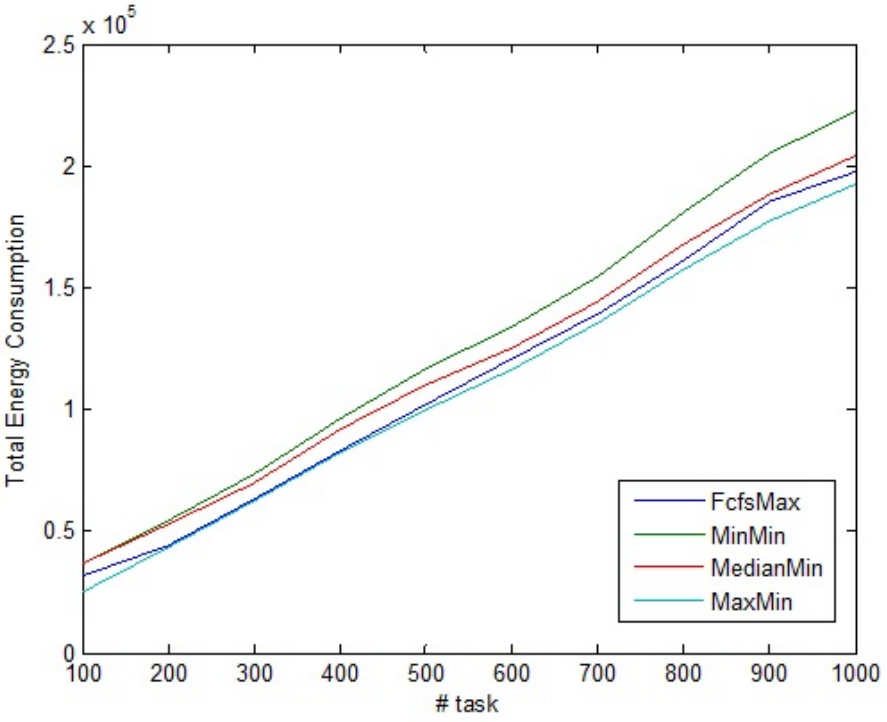


FIGURE 3.23: Energy consumption on 64 VMs

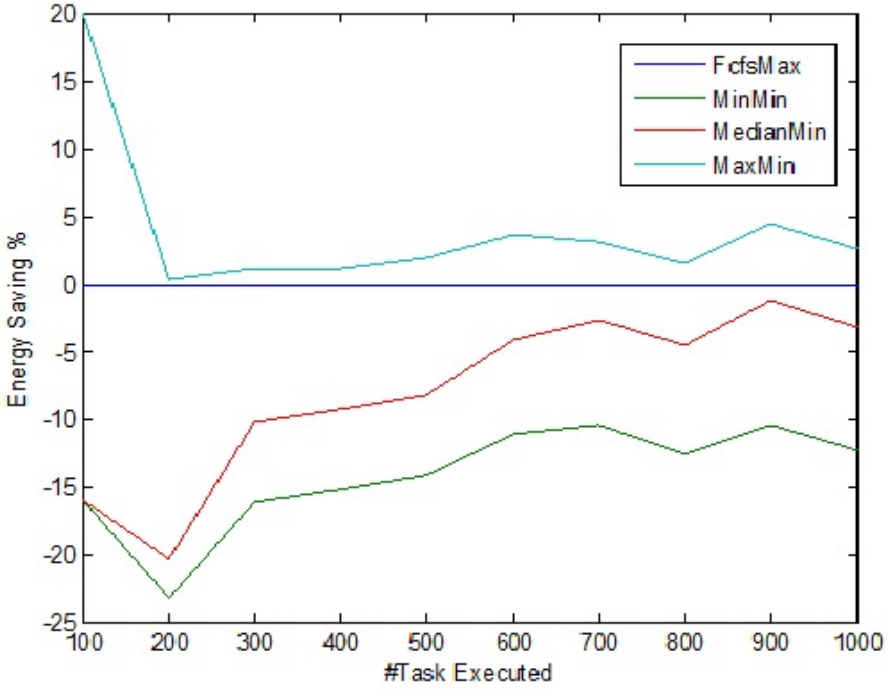


FIGURE 3.24: Energy saving on 64 VMs

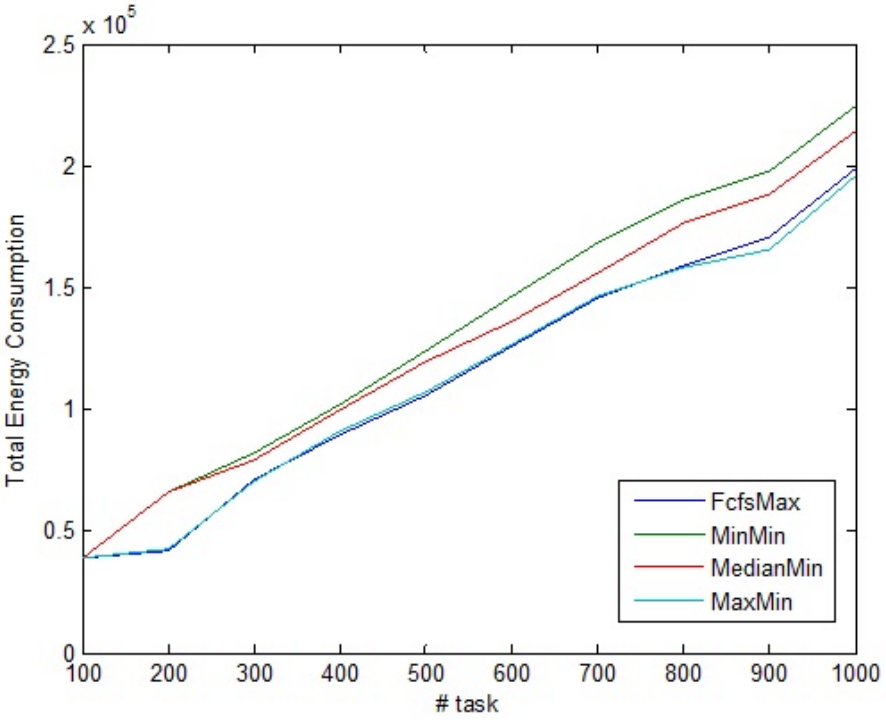


FIGURE 3.25: Energy consumption on 128 VMs

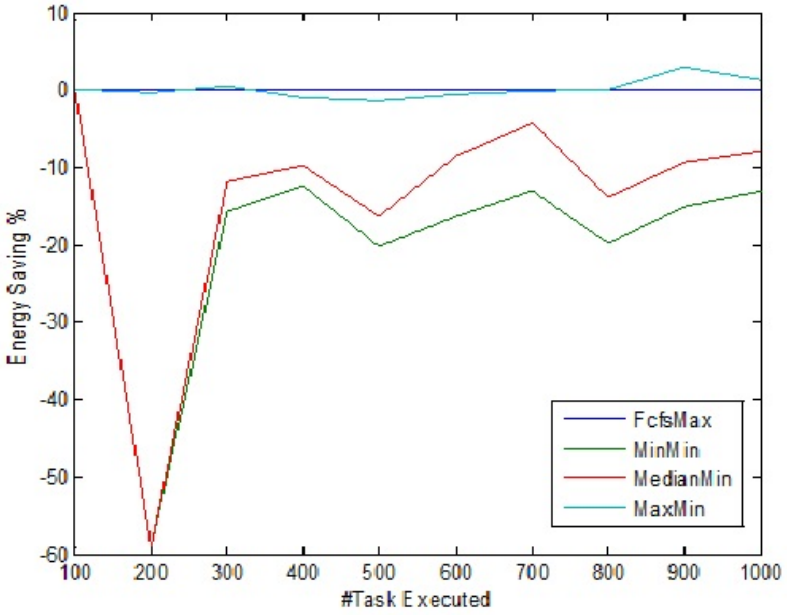


FIGURE 3.26: Energy saving on 128 VMs

3.3.3.4 Conclusion : Observation-05

It is observed that the energy consumption of MaxMin scheduling is minimum in this group.

3.3.3.5 Observation-06

In this section we observe the energy consumption and energy saving of group-3 (see table 3.10) heuristic algorithms on 20, 40 and 60 VMs for 100 to 1000 tasks. The results are shown in Figure 3.27 to 3.34.

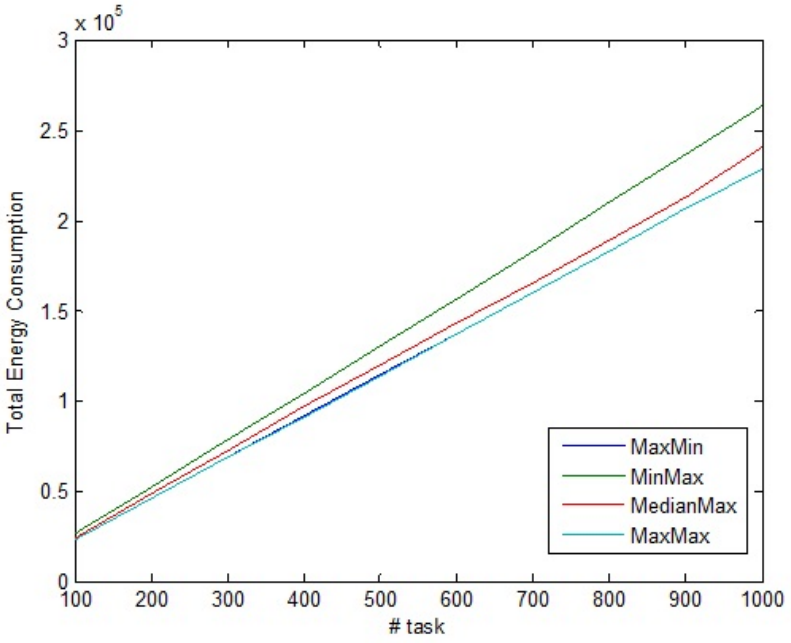


FIGURE 3.27: Energy consumption on 16 VMs

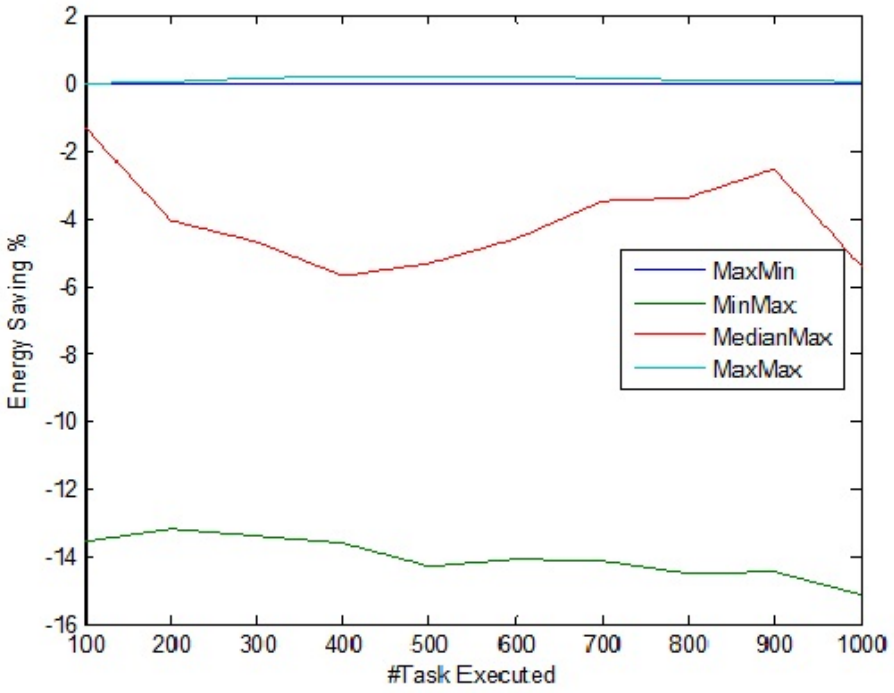


FIGURE 3.28: Energy saving on 16 VMs

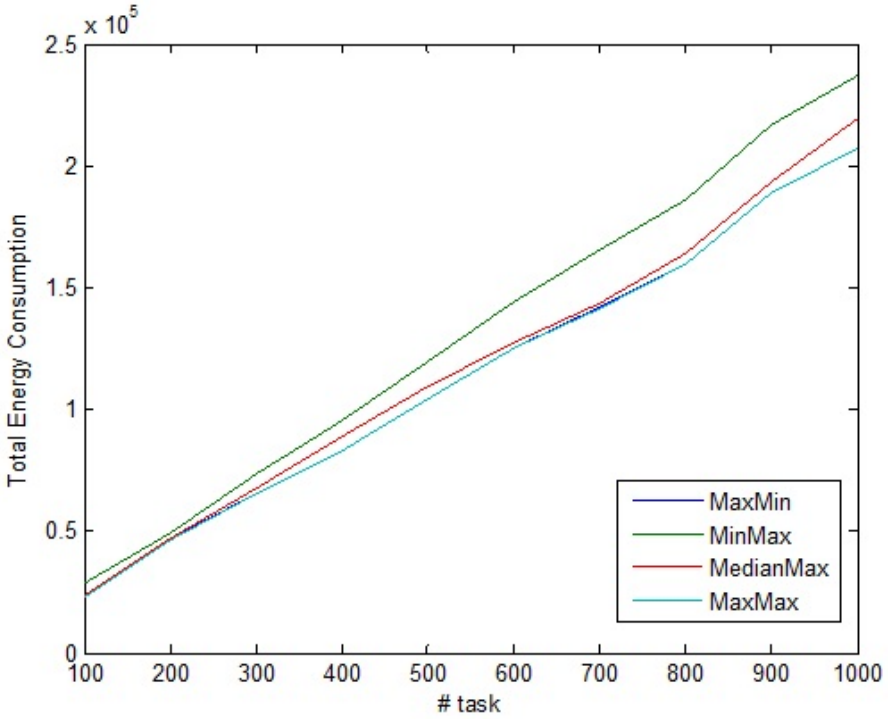


FIGURE 3.29: Energy consumption on 32 VMs

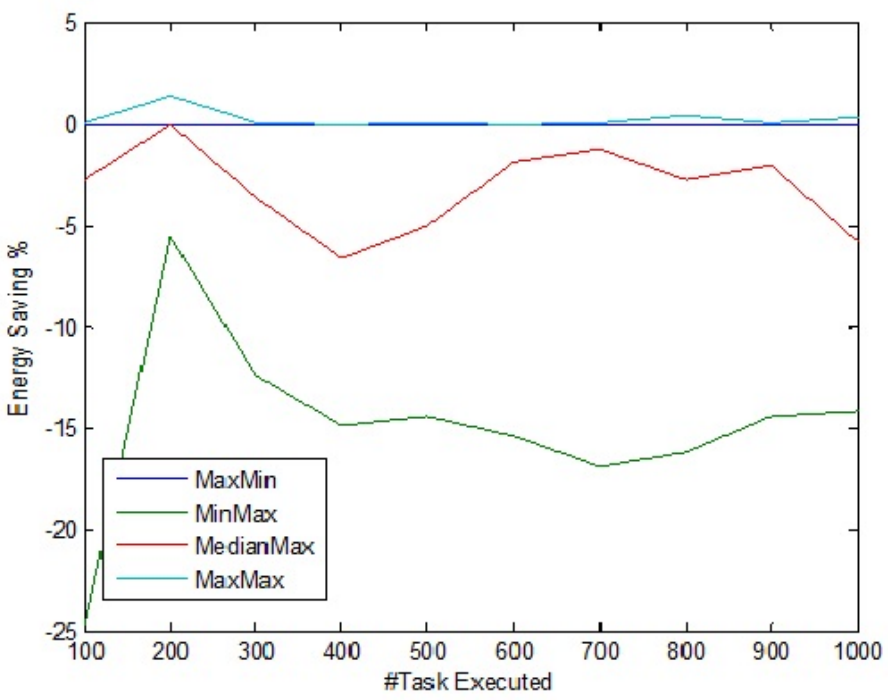


FIGURE 3.30: Energy saving on 32 VMs

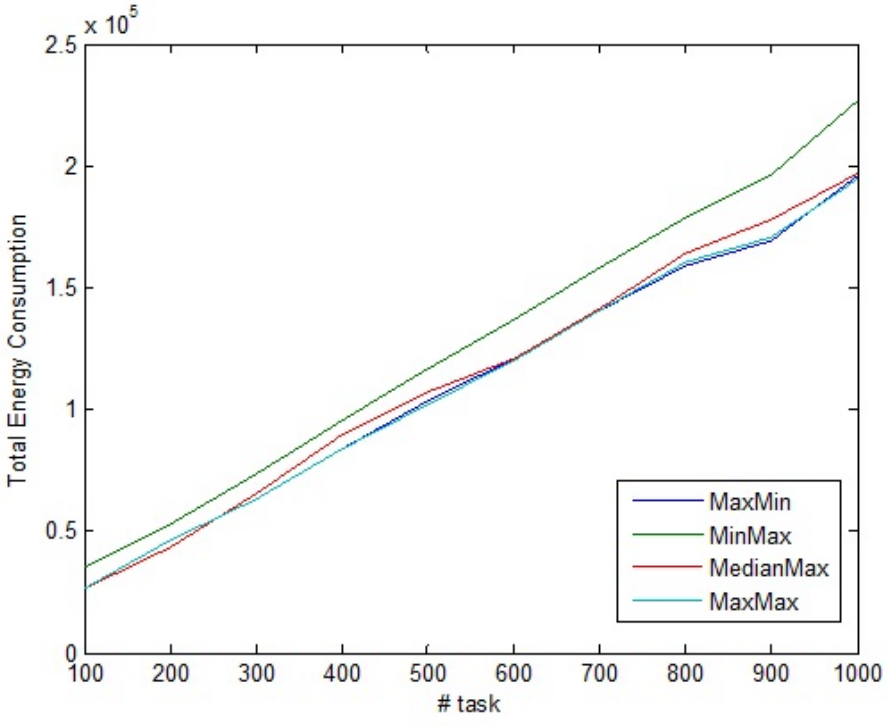


FIGURE 3.31: Energy consumption on 64 VMs

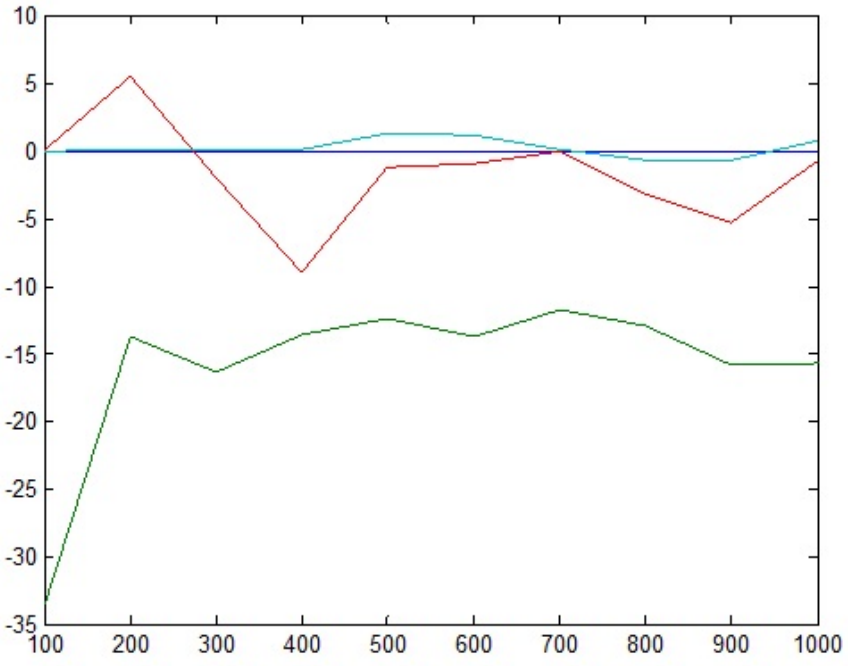


FIGURE 3.32: Energy saving on 64 VMs

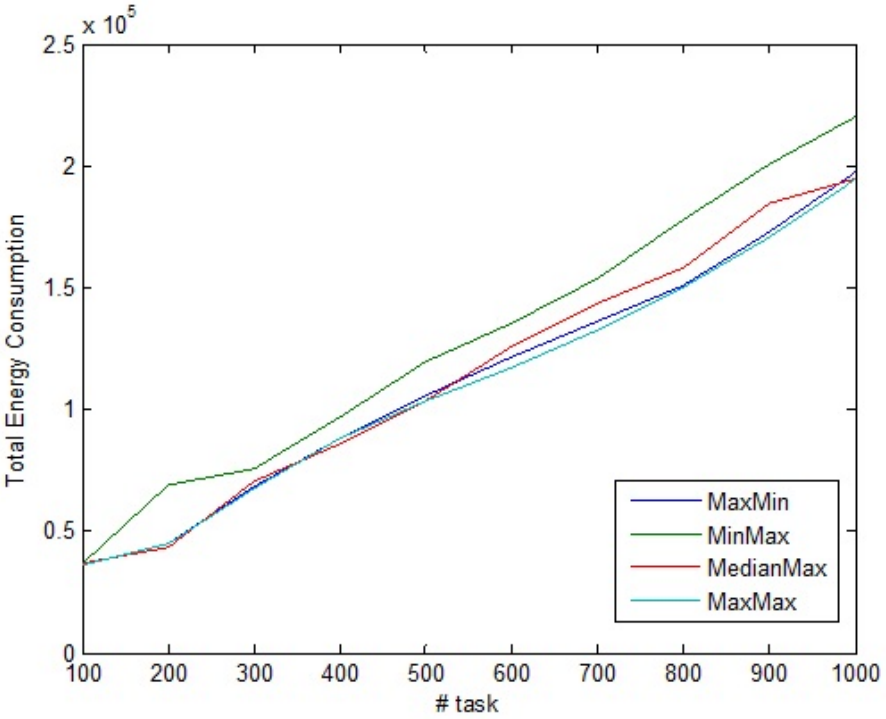


FIGURE 3.33: Energy consumption on 128 VMs

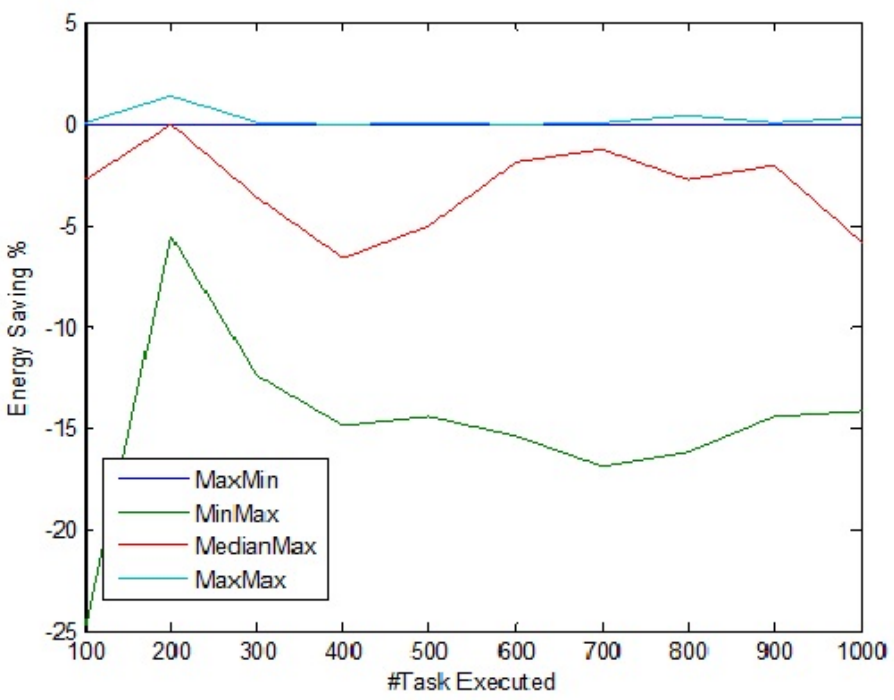


FIGURE 3.34: Energy saving on 128 VMs

3.3.3.6 Conclusion: Observation-06

It is observed that the energy consumption of MaxMax scheduling is minimum in this group.

3.3.4 Observation Scenario-2: Percentage of Energy Saving

In this section, we have observer the percentage of energy saving of ten different greedy heuristic algorithms compared to the FcfsRand. The simulation results for percentage of energy saving for 5000 tasks on 128 VMs are presented in Figure 3.35. The maximum energy saved is 11.5% by MaxMax compared to FcfsRand.

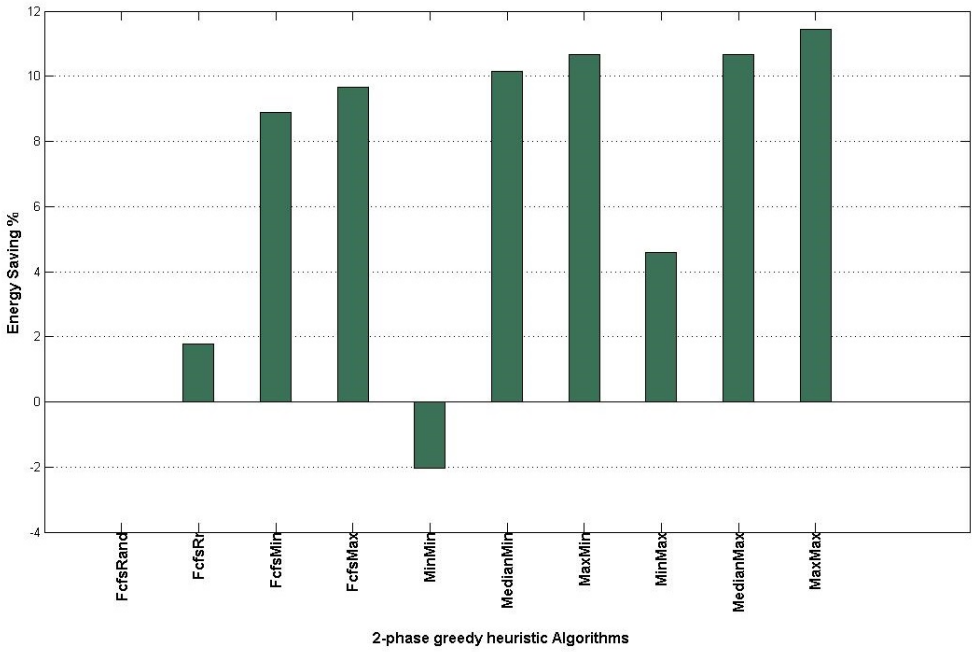


FIGURE 3.35: Energy saving comparison

3.4 Conclusions

Simulation experiments were conducted to examine the performance of simple heuristic based task consolidation algorithms to optimize the energy consumption in cloud computing system. An average case analysis is presented for ten heuristic different task consolidation algorithm with inconsistent ETC matrix. Simulation results proves the most of the time the performance of *MaxMax* scheduling outperform among 2-phase ten different heuristic algorithms. The maximum energy saved is 11.5% by MaxMax compared to FcfsRand.

CHAPTER 4

Energy Efficient Task Consolidation using Genetic Algorithm

Introduction

Genetic Algorithm Based Task Scheduling

Simulation Results

Conclusions

Chapter 4

Energy Efficient Task Consolidation using Genetic Algorithm

4.1 Introduction

In cloud, processing loads arrive from many users at random time instants in the form of task. A proper resource allocation policy attempts to assign this task to available VMs on different host so to complete the execution of the tasks in the shortest possible time with minimum power consumption. The complexity of the resource allocation problem with cloud increases with the number of hosts and becomes difficult to solve effectively. The resource allocation problem is a combinatorial problem and known to be NP-complete. The exponential solution space of the load balancing problem can be searched using heuristic techniques based on Genetic algorithms to obtain sub-optimal solution in acceptable time [40, 4]. The genetic algorithm is an evolutionary algorithm, that have been proven to be a successful in generating sub-optimal solutions to many scheduling problems A genetic algorithm performs a multi-directional search by maintaining a population

of potential solutions and an objective (evaluation) function which plays the role of an environmental [10, 22].

4.2 Genetic Algorithm Based Task Scheduling

In this section, we propose the GA-based task scheduling model and introduces a suitable codification scheme for chromosome. We also explain how making an optimal task schedule and compose elements of the GA scheduling function. To generate a new population, we have generated a POP_SIZE number of random initial population and calculating the fitness value of individuals. Then, using roulette wheel selection method, parents are selected to produce offsprings using single point crossover with probability 0.8. Some of the individuals are subjected to the mutation with a probability 0.2. The population for the next generation are selected again through roulette wheel selection method. The constant population size has been maintained for a fixed number of iterations. The individual from the last generation with minimum energy value is selected to allocate the tasks to VMs.

4.2.1 A Genetic Algorithm

The algorithm 5 described in this section is straightforward with two parts: initialization and looping. After initialization, it generates the feasible solution randomly, and then find the fitness value for best solutions. In looping parts, it checks whether the termination condition is met. If looping continues, selection, crossover (algorithm 8) and mutation (algorithm 9) operators are applied in a sequence. Then the better solution is saved during this iteration. At the end of the program, the saved best solution will be output as the optimized result.

Algorithm 5 workflow of a Genetic Algorithm

1: Find the fitness value of each chromosome in the population.
2: Reproduce a new population by repeating the following steps.
3: Select two individual chromosomes from a population according to selection method, roulette wheel selection.
4: Cross over the selected parents if crossover probability met, to produce a new child. Otherwise, the children are an exact copy of parents.
5: Mutate each new offspring (child) if a mutation probability met, at each locus (position in the chromosome).
6: Place new child in a reproduced population.
7: Store Best individual solution.
8: If the Maximum number of generation reached, stop, and return the best solution, else Go to step 2.

4.2.2 Encoding

A chromosome in this GA consists of $|C|$ genes, each represents the allocated resource ID (VM ID) to the task. The value of a gene is a positive integer between 1 and VM_MAX, representing the virtual machine where the task is allocated. Figure 4.1 shows an example of task scheduling and its corresponding chromosome. In this example, task t1, t2, t3, t4, t5, t6, t7, t8, t9 and t10 is placed on VM1, VM1, VM4, VM1, VM4, VM3, VM3, VM2, VM4, and VM4 respectively.

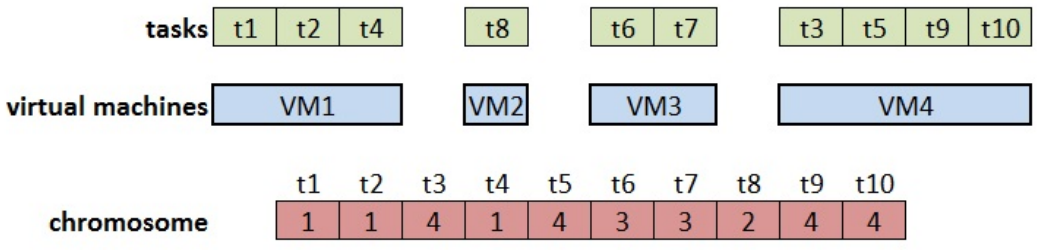


FIGURE 4.1: Individual Encoding(chromosome)

4.2.3 Fitness Function

This Fitness function finds the makespan of giving task execution pattern.

Algorithm 6 Fitness Function Algorithm

Input: Task sequence and ETC Matrix

Output: Makespan

1: initialize makespan(R^*) = 0

2: for each resource R_j find the makespan using equation 4.1

$$makespan(R_j) = \sum_{i=1}^{n} ETE_{(i,j)} \tag{4.1}$$

where R_j is j^th resource and i is task id.

3: return MAX(R^*)

4.2.4 Initial Population

In this thesis, an initial population of individuals is generated randomly using the algorithm 7

Algorithm 7 Generation of initial population algorithm

Input: population size(popsize), chomosome length(chlength)

Output: initial population,P

1: **for** $j = 1$ to *popsize* **do**

2: **for** $i = 1$ to *chlength* **do**

3: p(j,i) = round(random()*VM_MAX)

4: **end for**

5: **end for**

6: return P

4.2.5 Selection

In our GA, the roulette wheel selection method is used to select the population for the reproduction of the next generation.

4.2.6 Crossover

Our GA adopts a midpoint crossover (single point) operator with crossover probability 0.8, which is described in algorithm 8. In fig 4.2 show the chromosomes of parents and children before and after crossover respectively.

Algorithm 8 Mid-Point uniform crossover(Single Point) Algorithm

Input: two parent chromosome,C1,C2

Output: two child chromosome,CC1,CC2

1: cl ← length(C1)

2: crossover point, cp = cl/2

3: CC1 ← C1(1 : cp) U C2(cp : cl)

4: CC2 ← C1(cp : cl) U C2(1 : cp)

5: return CC1,CC2

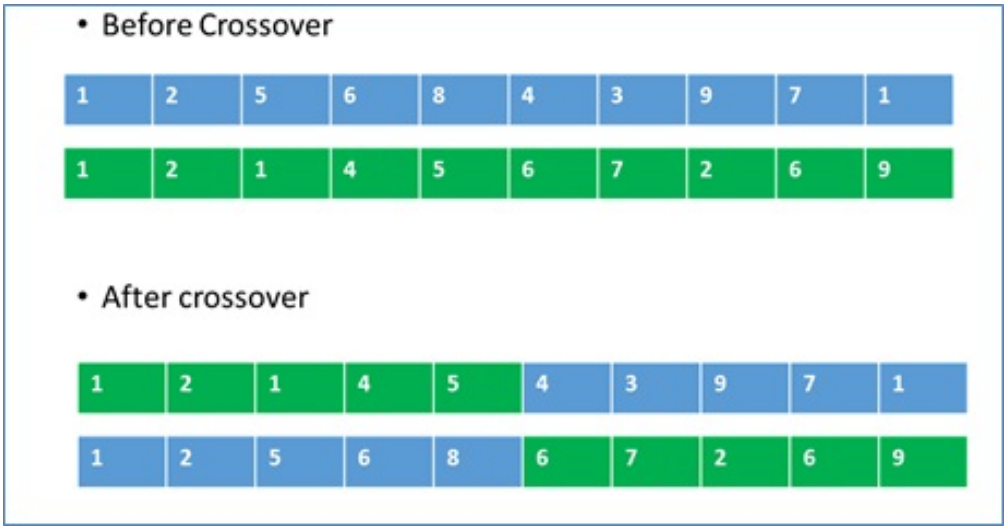


FIGURE 4.2: Example of mid-point crossover(single point)

4.2.7 Mutation

The mutation operator randomly picks up a gene in the chromosome and inverts the value of the chosen gene. Algorithm 4.3 shows how the mutation operator works. Constraints 1) make sure that each task will be assigned to one and only one virtual machine; constraints 2) guarantee that the total CPU workload on the VM_j will not exceed the maximum utilization capacity. In fig4.3 show an example, task t5 initially allocated to VM5, mutated to VM6.

Algorithm 9 Mutation at random point with 0.2 mutation probability

Input: a chromosome,C

Output: a mutated chromosome, CM

1: CM ← C
2: randomly generate a task id i, where 1 ≤ i ≤ $|C|$
3: randomly generate a real value between 0 and 1, *mp*
4: **if** (mp < 0.5) **then**
5: randomly generate a virtual machine j, where 1 ≤ j ≤ VM_MAX
6: replace CM(i) ← j
7: **end if**
8: output CM

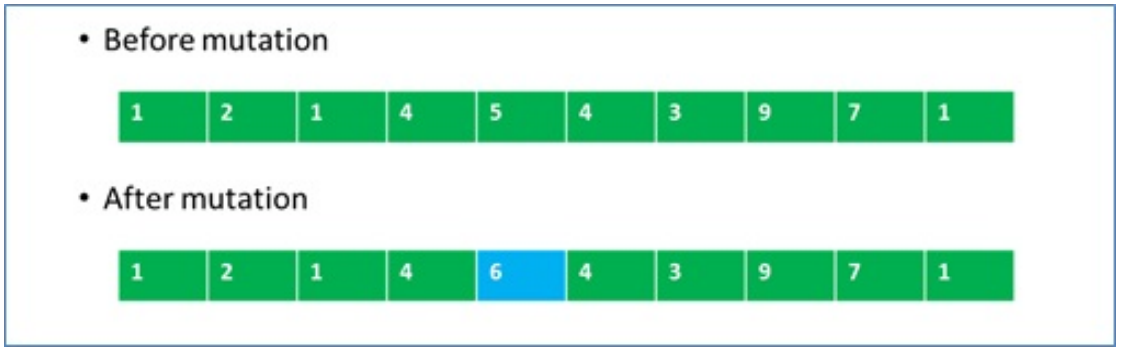
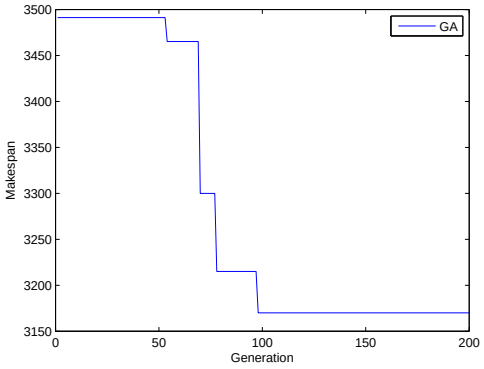


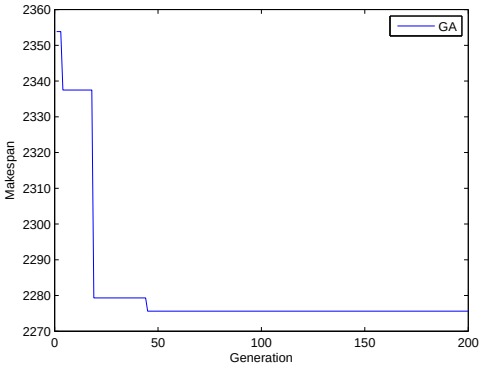
FIGURE 4.3: Example of mutation at random point with 0.2 mutation probability

4.2.8 Stopping condition

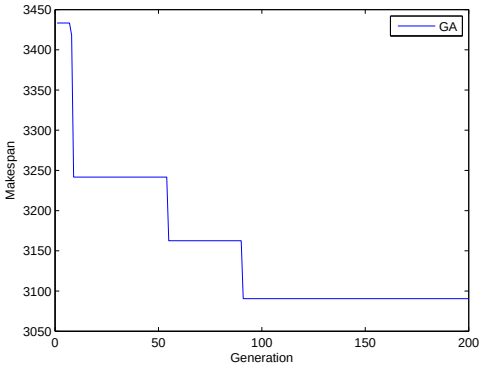
We have done the following simulation experiments for 500 tasks on 50 VMs, initial population size is 500, mutation probability is 0.2 and the single point crossover with crossover probability 0.8 to decide the stopping criteria. Figure 4.4a, 4.4b, and 4.4c show that the optimal result can be found after the 100 generation. In our GA, the stopping condition is decided by the maximum number of generations(MAX_GEN), which is equal to 100.



(A) Generation vs fitness level



(B) Generation vs fitness level



(C) Gneration vs fitness level

FIGURE 4.4: No of Generation vs fitness level for deciding optimal stopping condition

4.3 Simulation Results

In this section, we simulated our experiments using the discrete event system modeling [22] for the genetic algorithm based task scheduling and conducted the

various experiments. We have also compared those results with the random and RR scheduling model. The following parameters are taken in our simulation experiments: initial population size is 500, number of VMs are 50 and 100, stopping condition is 100 generations. A total 2500 tasks were generated using the ETC Model proposed by Zomaya [18]. The figure 4.5 and figure 4.6, shows the experimental result of Random scheduling, RR scheduling and GA based scheduling on 50 VMs and 100 VMs respectively.

initial population size = 500

number of VMs =50

generation = 100

stopping condition =100 generations

no of tasks = 2500 (generated using the ETC model)

crossover = single point crossover with 0.8 probalility

mutation = random point mutation with 0.2 probalility

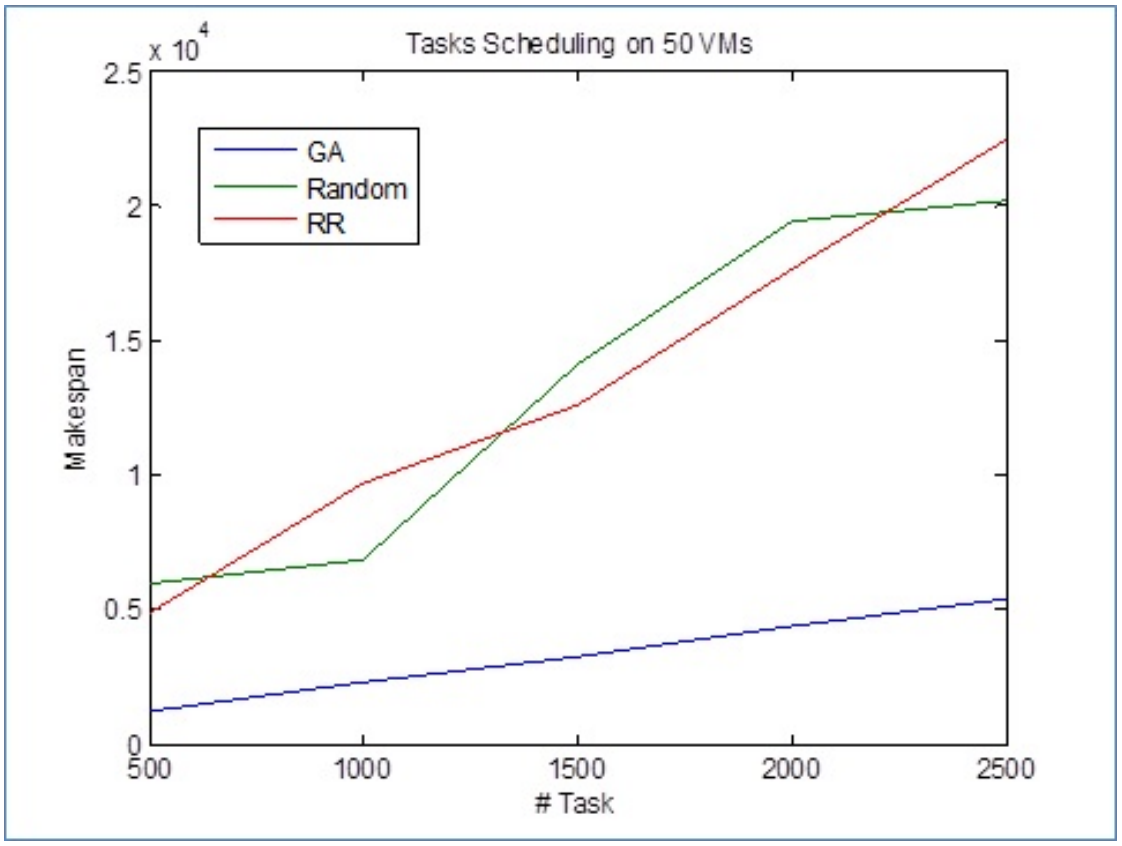


FIGURE 4.5: Task scheduling on 50 VMs in cloud computing infrastructure.

initial population size = 500

number of VMs =100

generation = 100

stopping condition =100 generations

no of tasks = 2500 (generated using the ETC model)

crossover = single point crossover with 0.8 probalility

mutation = random point mutation with 0.2 probalility

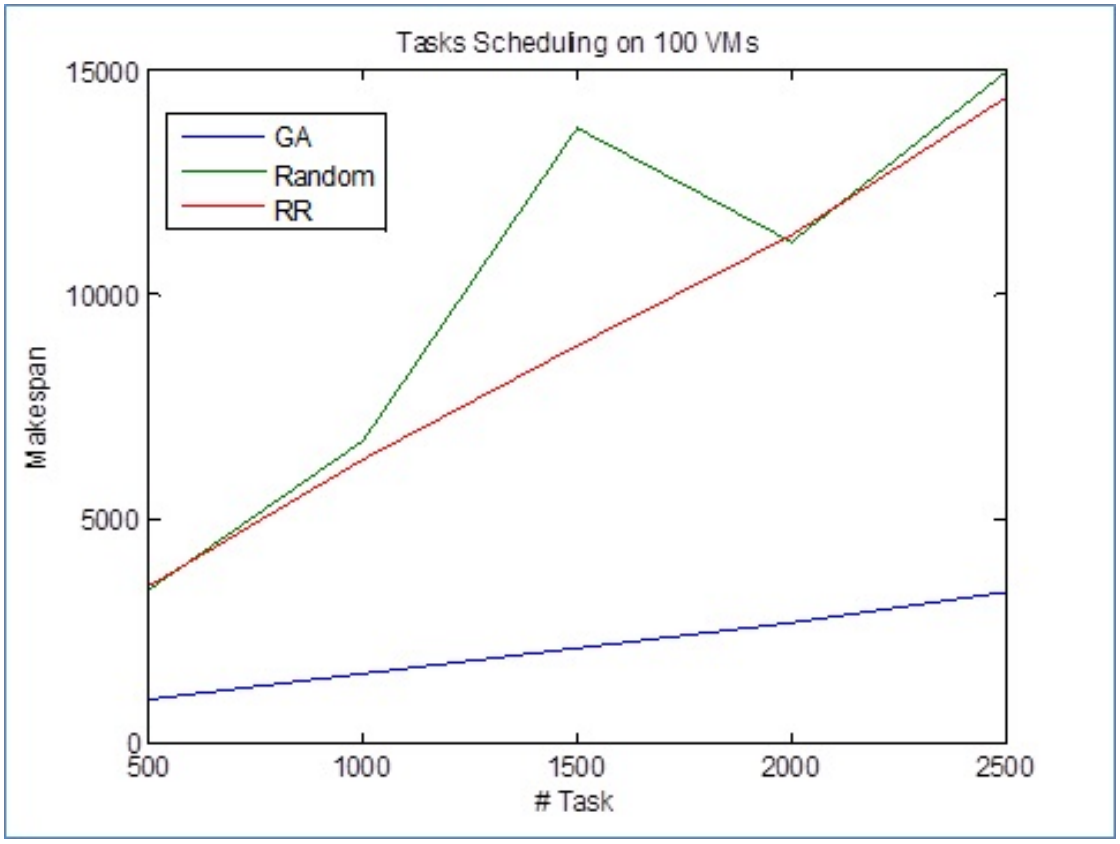


FIGURE 4.6: Task scheduling on 100 VMs in cloud computing infrastructure.

4.4 Conclusions

The experimental results (figure 4.5 and figure 4.6) show that the GA based scheduling model outperforms the existing Rondom and RR scheduling models.

CHAPTER 5

Conclusions and Future Works

Conclusions

Future Works

Chapter 5

Conclusions and Future Works

5.1 Conclusions

The resource allocation problem in a cloud computing environment has been addressed as an optimization problem. We formulated the resource allocation problem as Linear Programming Problem to optimize the energy consumption in cloud computing infrastructure. Heuristics and meta-heuristic technique are preferred by the researchers to address NP-complete problem. We have used the greedy algorithms for resource allocation problem. Simulation experiments were conducted to examine the performance of ten heuristic based resource allocation algorithms to optimize the energy consumption in cloud computing system. An average case analysis is presented for ten heuristic different resource allocation algorithm with inconsistent ETC matrix. Simulation results prove the *MaxMax* heuristic algorithm is preferred over others.

We have proposed a new simple genetic algorithm frame work for Energy Efficient Task Consolidation in cloud computing enviourment. The proposed genetic algorithm uses the fixed number of iteration to obtained the suboptimal solution for task allocation problem. The same has been compared with existing Rondom and Round Robin scheduling statgies using in house simulator. The simulation experimental results indicated in the favour of the GA based resource algorithm.

5.2 Future Works

More applications and services over the internet are available through the Cloud computing platforms to meet the user requirement. It becomes a trend to have an application to be hosted on the cloud. The scalability of a host becoming essential to provide support to these applications through virtualization. Resource allocation problem always meets the new challenges to meet the scalability, and service level agreements. There is a need to study the performance of the proposed heuristics can be analyzed for scalability and fault tolerance of the hosts. Scope of using evolutionary algorithms like Particle swarm optimization (PSO), Ant colony optimization (ACO), and Simulated annealing (SA) may be investigated in resource allocation in cloud computing. Effective and reliability of services become a challenge in a cloud computing environment. Autonomic computing systems are designed to exhibit the ability of self-monitoring, self-repairing, and self-optimizing. There is a scope to design autonomic resource allocation strategy to meet the SLA for user requirements[15].

Appendix A

Dissemination of Work

Published:

1. **Dilip Kumar** and Bibhudatta Sahoo, Energy Efficient Heuristic Resource Allocation for Cloud Computing, International Journal of Artificial Intelligent Systems and Machine Learning, CIIT, vol.6 no 1, January 2014, pp.32-38.

2. Bibhudatta Sahoo, **Dilip Kumar**, and Sanjay Kumar Jena. "Analysing the Impact of Heterogeneity with Greedy Resource Allocation Algorithms for Dynamic Load Balancing in Heterogeneous Distributed Computing System."International Journal of Computer Applications 62 (2013).

3. Bibhudatta Sahoo, **Dilip Kumar**, and Sanjay Kumar Jena. "Observing the Performance of Greedy algorithms for dynamic load balancing in Heterogeneous Distributed Computing System." 1st International Conference on Computing, Communication and Sensor Networks-CCSN,(2012), vol. 62, PIET, Rourkela, Odisha, pp. 265-269, 2012.. 2012.

Bibliography

[1] Ali, S., Siegel, H. J., Maheswaran, M., and Hensgen, D. (2000). Task execution time modeling for heterogeneous computing systems. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 185–199. IEEE.

[2] Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768.

[3] Black, P. E. (2005). Greedy algorithm. *Dictionary of Algorithms and Data Structures*.

[4] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., et al. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing*, 61(6):810–837.

[5] Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*.

[6] Calheiros, R. N., Buyya, R., and De Rose, C. A. (2009). A heuristic for mapping virtual machines and links in emulation testbeds. In *Parallel Processing, 2009. ICPP'09. International Conference on*, pages 518–525. IEEE.

[7] Cardosa, M., Korupolu, M. R., and Singh, A. (2009). Shares and utilities based power consolidation in virtualized server environments. In *Integrated*

Network Management, 2009. IM'09. IFIP/IEEE International Symposium on, pages 327–334. IEEE.

[8] Chedid, W., Yu, C., and Lee, B. (2005). Power analysis and optimization techniques for energy efficient computer systems. *Advances in Computers*, 63:129–164.

[9] Galloway, J. M., Smith, K. L., and Vrbsky, S. S. (2011). Power aware load balancing for cloud computing. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 19–21.

[10] Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

[11] Gray, L., Kumar, A., and Li, H. (2008). Characterization of specpower_ssj2008 benchmark. In *SPEC Benchmark Workshop*.

[12] Hwang, K., Fox, G., and Dongarra, J. (2012). *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kaufmann.

[13] Jing, S.-Y., Ali, S., She, K., and Zhong, Y. (2013). State-of-the-art research study for green cloud computing. *The Journal of Supercomputing*, pages 1–24.

[14] Kang, Q.-M., He, H., Song, H.-M., and Deng, R. (2010). Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization. *Journal of Systems and Software*, 83(11):2165–2174.

[15] Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

[16] Kumar, D. and Sahoo, B. (2014). Energy efficient heuristic resource allocation for cloud computing. *International Journal of Artificial Intelligent Systems and Machine Learning*, 6(1):32–38.

[17] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. *Cluster computing*, 12(1):1–15.

[18] Lee, Y. C. and Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280.

[19] Liu, L., Wang, H., Liu, X., Jin, X., He, W. B., Wang, Q. B., and Chen, Y. (2009). Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38. ACM.

[20] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., and Chompoo-inwai, C. (2007). An ant colony optimization for dynamic job scheduling in grid environment. *International Journal of Computer & Information Science & Engineering*, 1(4).

[21] Mell, P. and Grance, T. (2011). The nist definition of cloud computing (draft). *NIST special publication*, 800(145):7.

[22] Michalewicz, Z. (1996). *Genetic algorithms+ data structures= evolution programs*. springer.

[23] Minas, L. and Ellison, B. (2009). The problem of power consumption in servers. *Intel Corporation. Dr. Dobb's*.

[24] Mukherjee, K. and Sahoo, G. (2009). Mathematical model of cloud computing framework using fuzzy bee colony optimization technique. In *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*, pages 664–668. IEEE.

[25] Ohmae, H., Ikkai, Y., Komoda, N., Horiuchi, K., and Hamamoto, H. (2003). A high speed scheduling method by analyzing job flexibility and taboo search for a large-scale job shop problem with group constraints. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, volume 2, pages 227–232. IEEE.

[26] Packard, H. (2011). Power regulator for proliant servers.

[27] Rodero, I., Jaramillo, J., Quiroz, A., Parashar, M., Guim, F., and Poole, S. (2010). Energy-efficient application-aware online provisioning for virtualized

clouds and data centers. In *Green Computing Conference, 2010 International*, pages 31–45. IEEE.

[28] Rosenblum, M. (2004). The reincarnation of virtual machines. *Queue*, 2(5):34.

[29] Sahoo, B., Kumar, D., and Jena, S. K. Observing the performance of greedy algorithms for dynamic load balancing in heterogeneous distributed computing system. In *1st International Conference on Computing, Communication and Sensor Networks-CCSN*.

[30] Sahoo, B., Kumar, D., and Kumar Jena, S. (2013). Analysing the impact of heterogeneity with greedy resource allocation algorithms for dynamic load balancing in heterogeneous distributed computing system. *International Journal of Computer Applications*, 62.

[31] Song, Y., Wang, H., Li, Y., Feng, B., and Sun, Y. (2009). Multi-tiered on-demand resource scheduling for vm-based data center. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 148–155. IEEE Computer Society.

[32] Srikantaiah, S., Kansal, A., and Zhao, F. (2008). Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10. USENIX Association.

[33] ssj2008, S. (2013). Benchmark results summary of hitachi model no: Ha8000/ss10 (dl2).

[34] STANDARDIZATION and ITU (2012). Fg cloud technical report v1.0. *International Telecommunication Union*.

[35] Venkatachalam, V. and Franz, M. (2005). Power reduction techniques for microprocessor systems. *ACM Computing Surveys (CSUR)*, 37(3):195–237.

[36] Verma, A., Ahuja, P., and Neogi, A. (2008). pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware 2008*, pages 243–264. Springer.

[37] VmWare (2012). Vmware.

[38] Ye, K., Huang, D., Jiang, X., Chen, H., and Wu, S. (2010). Virtual machine based energy-efficient data center architecture for cloud computing: a performance perspective. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 171–178. IEEE Computer Society.

[39] Yeo, C. S. and Buyya, R. (2006). A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience*, 36(13):1381–1419.

[40] Zomaya, A. Y. and Teh, Y.-H. (2001). Observations on using genetic algorithms for dynamic load-balancing. *Parallel and Distributed Systems, IEEE Transactions on*, 12(9):899–911.