Dynamic Virtual Machine Placement in Cloud Computing

Arnab Kumar Paul

(Roll No: 213CS3190)

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

May, 2015.

Dynamic Virtual Machine Placement in Cloud Computing

Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

by

Arnab Kumar Paul

(Roll No: 213CS3190)

under the guidance of

Dr. Bibhudatta Sahoo

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

May, 2015.

Dedicated to my Parents and Siblings

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela - 769008, Odisha, India

CERTIFICATE

This is to certify that the work in the thesis entitled ***Dynamic Virtual Machine Placement in Cloud Computing*** submitted by ***Arnab Kumar Paul*** 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, National Institute of Technology, Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Bibhudatta Sahoo
Assistant Professor
Department of Computer Sc. and Engg.
National Institute of Technology
Rourkela-769008

Place: NIT, Rourkela
Date: 26-05-2015

Acknowledgment

First of all, I would want to express my sincere respect and thanks towards my supervisor Dr. Bibhudatta Sahoo, who has been the guiding light behind this work. I want to acknowledge him for introducing me to the exciting field of Cloud Computing and giving me the opportunity to work under his guidance. His undivided faith in this topic and ability to bring out the best of analytical and practical skills in people has been invaluable in tough periods. Without his invaluable suggestions and ever ready help it wouldn't have been possible for me to complete this thesis. I am extremely fortunate to have got a chance to work alongside such a wonderful person.

I express my gratitude towards all the faculty members of the CSE Department for their sympathetic cooperation.

During my studies at N.I.T. Rourkela, I made many friends. I would like to thank them all, for all the great moments I had with them.

When I look back at my accomplishments in life, I can see a clear trace of my family's concerns and devotion everywhere. My dearest mother, whom I owe everything I have achieved and whatever I have become; my beloved father, for always believing in me and inspiring me to dream big even at the toughest moments of my life; and my sister and brother-in-law; who were always my silent support during all the hardships of this endeavor and beyond.

Arnab Kumar Paul

213CS3190

Abstract

Cloud computing enables users to have access to resources on demand. This leads to an increased number of physical machines and data centers in order to fulfill the needs of users which are continuously on the increase. The increase in the number of active physical machines is directly proportional to the increase in the energy consumption. Thus, minimization of energy consumption has become one of the major challenges of cloud computing in recent years. There are many ways to power savings in data centers, but the most effective one is the optimal placement of virtual machines on physical machines. In this thesis, the problem of dynamic placement of virtual machines is solved in order to optimize the energy consumption. A cloud computing model is built along with the energy consumption model considering the states of physical machines and the energy consumption during live virtual machine migrations and the changes in the states of physical machines. The intelligent algorithms having a centralized approach, like genetic algorithm and simulated annealing algorithm have been used to solve the dynamic virtual machine placement problem in earlier research works but many unreachable solutions may result. Thus, a decentralized approach based on game theoretic method is used here in order to reach optimal solutions and also a list of executable live virtual machine migrations is provided to reach the optimal placement. In real world scenario, physical machines may or may not cooperate with each other to arrive at an optimal solution. Therefore, in this thesis both cooperative as well as non-cooperative game theoretic approaches have been used to find optimal solution to the dynamic virtual machine placement problem. It is seen that Nash equilibrium is achieved in polynomial time. The experimental results are compared with the results of best fit approach. Results show that energy consumption is minimized by modifying the placement of virtual machines dynamically.

Keywords - cloud computing; dynamic virtual machine placement; game theory; Nash equilibrium; energy consumption

Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Description
DC	Data Center
PM	Physical Machine
VM	Virtual Machine
SLA	Service Level Agreement
VMP	Virtual Machine Placement
QoS	Quality of Service
CSP	Cloud Service Provider
SP	Service Provider
SaaS	Software as a Service
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
VMM	Virtual Machine Monitor
CP	Constraint Programming
CCA	Canonical Correlation Analysis
SIP	Stochastic Integer Programming
OVMP	Optimal Virtual Machine Placement
HDCF	Horizontal Dynamic Cloud Federation Platform

CHAPTER 1

INTRODUCTION

1.1 Introduction

Cloud computing acts as a model to enable users to have on demand access to computing resources with minimal management effort. It's emergence as a favored computing model to support large scale processing of huge volumes of data is commendable. Several multi-national organizations such as Google, Yahoo, Microsoft, Amazon and IBM have built cloud platforms for enterprises and users to access the cloud services.

Data Centers have been used to provide powerful computing resources for critical areas, such as nuclear physics, scientific simulation and geothermal experiments. A Data Center (DC) usually deploys a large number of Physical Machines (PMs) packed densely to maximize space utilization. Virtualization is one of the key concepts of data center management. The major advantage of virtualization is the possibility of running several operating system instances on a single PM thus utilizing the hardware capabilities more fully which allows administrators to save money on hardware and energy costs. In literature, these individual operating system instances are defined as Virtual Machines (VM). The computing resources of DC are made available to the users through VMs.

The VM scheduling in a cloud computing environment is very crucial as the number of users continuously increase. The VM scheduling algorithm greatly affects the performance of the whole system and its throughput. The placement of VMs to DCs is called Virtual Machine Placement (VMP) over DCs and the placement of VMs to PMs is called VMP over PMs. The effectiveness of VMP is related to a Quality of Service (QoS) as per the services. The objective of VMP will be a minimum number of data centers with a much larger per-data center utilization. More availability and flexibility of DCs is achieved, while the operational costs and hardware expenses such as physical space, energy etc. are reduced.

In this thesis, Dynamic Virtual Machine Placement over Physical machines is considered for user/broker requests. A decentralized approach is used for decision making and the existence of Nash Equilibrium is proved.

1.1.1 Anatomy of Cloud Computing

In cloud computing, virtualization acts as a key player in providing scalable and dynamic architectures. Along with resource sharing and scalability, virtualization provides the ability of virtual machine migration between physical servers for load balancing (20).

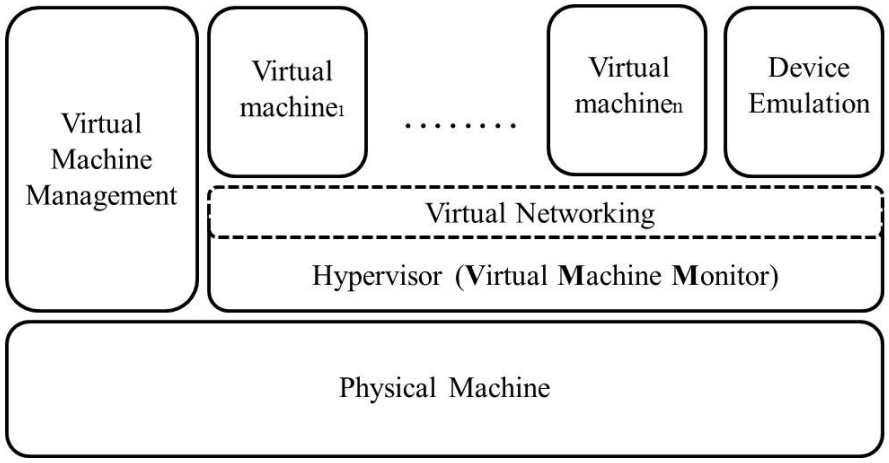


Figure 1.1: Core Elements of Cloud Node

Figure 1.1 shows the important elements in a node in cloud computing environment. Virtualization component in a node is provided by an infrastructure layer known as *Hypervisor* (also called as *Virtual Machine Monitor* [VMM]). This layer provides the interface to execute multiple instances of operating system at the same time on a single PM. The hypervisor creates objects known as *Virtual Machines* which encapsulate operating system, configuration and applications. Device emulation is also provided to the physical machine either in hypervisor or as a VM. Virtual machine management takes place both locally in different physical machines and globally in a *Data Center* (DC).

The nodes represented in Figure 1.1 are then multiplied on a physical network with management orchestration over the entire infrastructure to form a *Data Center (DC)* as shown in Figure 1.2.

Figure 1.2: Data Center Representation

Hypervisors

Hypervisor acts as the base level of a physical machine (node). It manages the execution of the guest operating systems by providing them with a virtual operating platform. These are known as virtual machines. The virtual machines share the virtualized hardware resources of the node. The Linux Kernel Virtual Machine (KVM)is one of the best hypervisors which is also deployed in production environments.

Device Emulation

Hypervisors provide platform where VMs can share the virtualized physical resources. But in order to provide full virtualization, the whole node must be virtualized, which is the job of a device emulator. An example of a complete package (emulator and hypervisor) is QEMU.

Virtual Networking

Networking needs of a system intensifies as more and more VMs consolidate on physical servers. Thus, instead of VMs communicating on the physical level, the whole network

is also virtualized which reduces the load on the physical infrastructure. In order to optimally communicate between VMs, *virtual switches* are introduced. An example of a virtual network provider in cloud environment is *Open vSwitch*.

1.1.2 Cloud Computing Architecture

In a cloud computing environment, users depend on cloud service providers to fulfill their needs. Thus, some QoS parameters must be maintained by the cloud providers which are cataloged in the Service Level Agreement (SLA). In order to achieve this, market oriented architecture is needed instead of the traditional resource management architecture. Figure 1.3 showcases the cloud computing architecture which will support the market-oriented resource allocation.

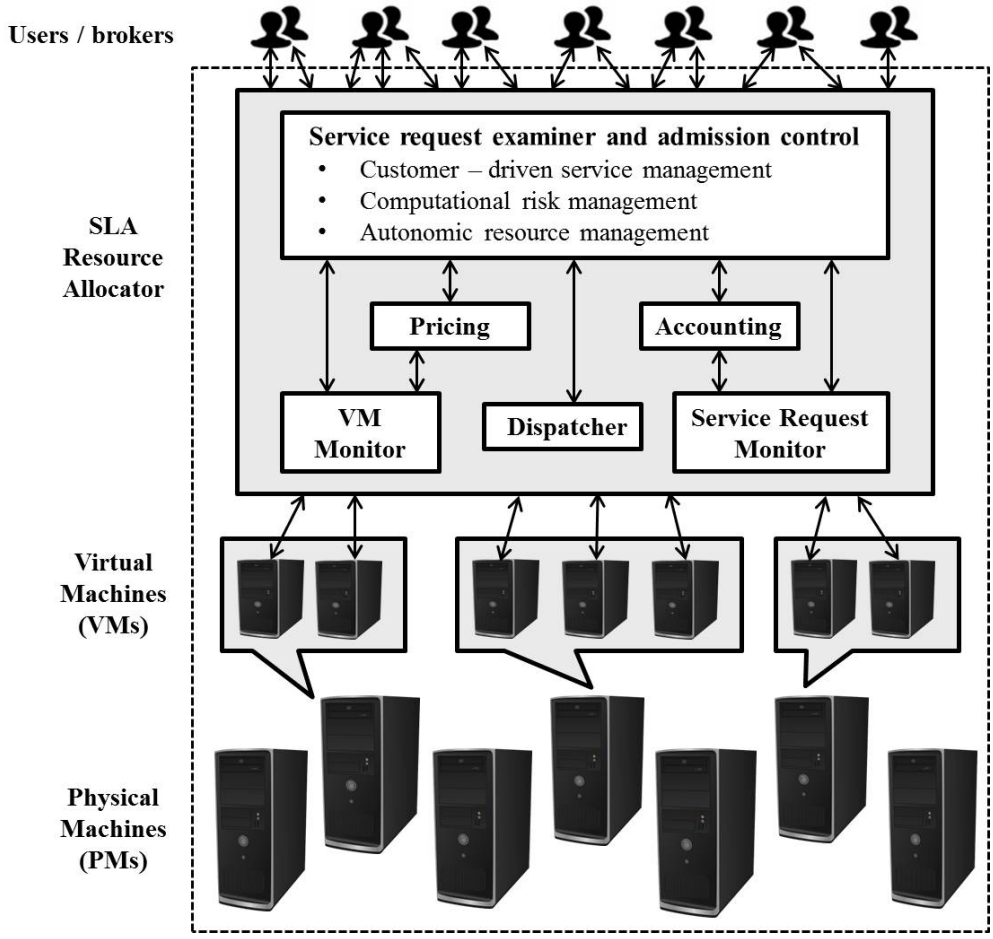


Figure 1.3: Cloud Computing Architecture

The cloud model is built using the following components.

Users or brokers which act on the behalf of users, submit their requests to the cloud data center in order to be processed.

SLA Resource Allocator provides the interface between the CSP/DC and the user/broker. On submission of a user request, the *Service Request Examiner* checks the request for QoS parameters' requirements to make a decision whether to accept or reject the user request. It obtains information from the *VM Monitor* on the availability of resources and already available workload from the *Service Request Monitor*, thus ensuring that no overloading occurs. Considering these parameters, the examiner assigns the user requests to VMs and decides the allotted VMs' resource requirements.

The *Pricing* operation makes decisions on the prices of service requests; that is whether to charge the request on the basis of time of submission, or resource availability or simply based on fixed rates. This mechanism aids in effective prioritization of allocation of resources in a data center.

Accounting operation is used to keep a tab on the actual resource consumption, according to which the pricing mechanism can calculate the final cost to be charged from users. It also helps in effective resource allocation decisions to be made by the service examiner by using the historical resource usage information kept by the accounting mechanism.

The *VM Monitor* keeps a check on the resource requirements and VMs' availability. *Dispatcher* starts executing the accepted user requests on the allotted VMs. *Service Request Monitor* keeps a tab on the status of the execution of the service requests.

1.2 Virtual Machine Placement in Cloud Computing

One of the major concepts in cloud computing is virtualization. It has the major advantage that it allows the execution of several instances of the operating system on a single PM, thus enabling complete utilization of the PM's hardware capacities. These instances of the operating system are called virtual machines. The placement of VMs in cloud computing environment is very crucial since the number of cloud users is on the rise. The scheduling of VMs greatly affects the whole system's performance and throughput. Virtualization in a physical machine is taken care of by *Hypervisor* as discussed in the previous section.

The virtual machine requests follow a two-tier distribution approach. A large number of *Physical Machines* are deployed in a *Data Centers*. A *Cloud Service Provider* can have multiple data centers. Thus the VM requests should be first distributed optimally over data centers. The requests in each data center are then distributed over physical machines.

The virtual machine placement can be said to be two types.

- Static virtual machine placement
- Dynamic virtual machine placement

1.2.1 Static Virtual Machine Placement

Static placement of VMs is done either during system startup or in offline mode. This is the initial placement of VMs in the cloud computing environment. No prior mapping of VMs is found. This type of VM placement does not consider either the states of the virtual machines and physical machines, or the arrival rate of the user requests.

Below is the diagrammatic explanation of the centralized model for static virtual machine placement following the two tier approach. First, VM placement over data centers and then the placement of requests over PMs in a particular data center are shown.

Figure 1.4 represents the flow of VM Requests from the users till they are distributed over data centers.

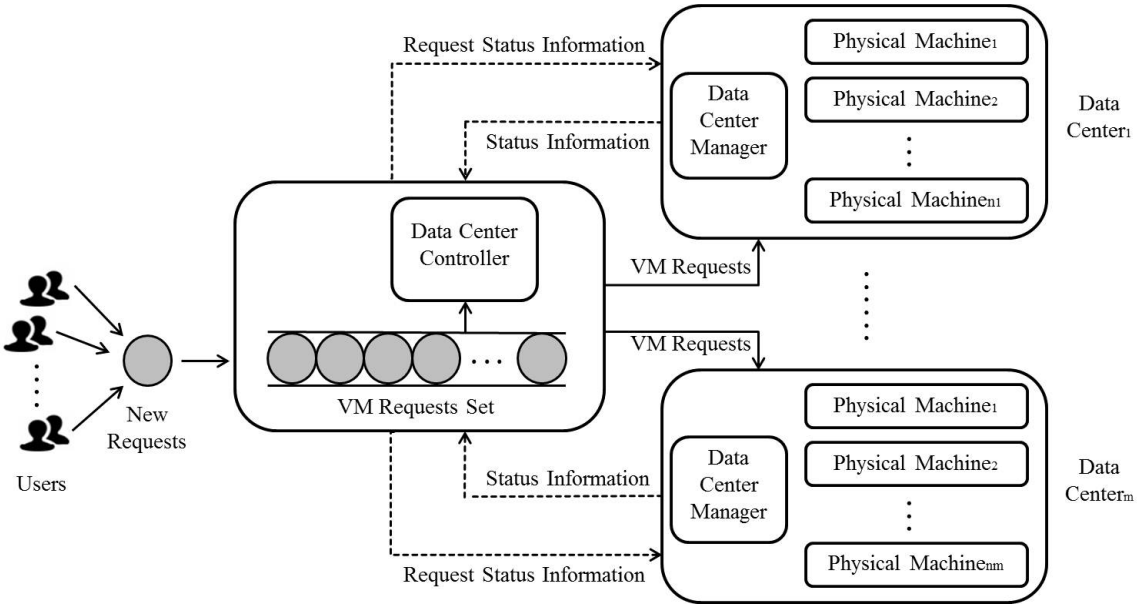


Figure 1.4: Static Virtual Machine Placement over Data Centers

Cloud users use the services provided by the cloud service provider and issue requests. These requests are in the form of virtual machine requests since every request will be completed on a virtual machine on top of a physical machine. The VM requests comprise the VM Requests Set. Every data center has multiple physical machines and a *Data Center Manager* to control all the PMs. In order to control all the data centers, a *Data Center Controller* is used.

The data center controller receives the VM requests set. It then requests status information from all the data center managers which have the information of their respective data centers. The VM requests contain information of the resources (CPU, RAM, Network etc.) needed in order to complete the request. The data center managers send information of the available resources to the controller. The data center controller optimally distributes VM requests to the data center managers following a heuristic algorithm. This approach to distribute VM requests is a *Centralized Approach*, where the decision to schedule requests depends on the central controller.

Figure 1.5 depicts the static placement of virtual machine requests in physical machines in a single data center.

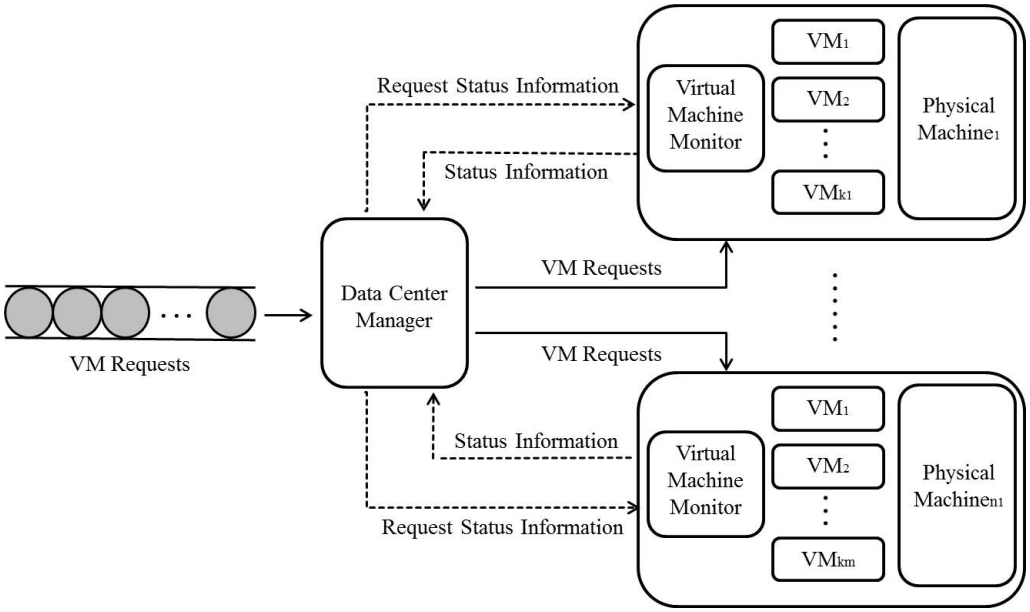


Figure 1.5: Static Virtual Machine Placement over Physical Machines

In a data center, there are multiple physical machines on top of which lies the *Virtual Machine Monitor [VMM] (Hypervisor)*. The VMM has the responsibility for virtualization in PMs. The data center manager sends requests to the VMMs to provide the status information of all the physical machines. The data center manager already has the VM requests' information. Upon arrival of the status information from the VMMs, the DC manager places the VM requests to the individual physical machines in order to process them and send back the response to the cloud users. The placement of VM requests over physical machines is also a centralized approach as the decision is taken by a central authority, in this case the data center manager.

1.2.2 Dynamic Virtual Machine Placement

If an existing mapping of VMs onto PMs is present, we go for dynamic placement of virtual machines. The main goal of dynamic VM placement is to achieve optimum solutions from the already present mapping of VMs at minimal cost. The optimality parameters may vary from minimization of the response time to the minimization of energy consumption or a combination of multiple parameters. The rate of arrival of user requests as well as the states of both VMs and PMs need to be considered while taking a decision. Below is an example of dynamic virtual machine placement.

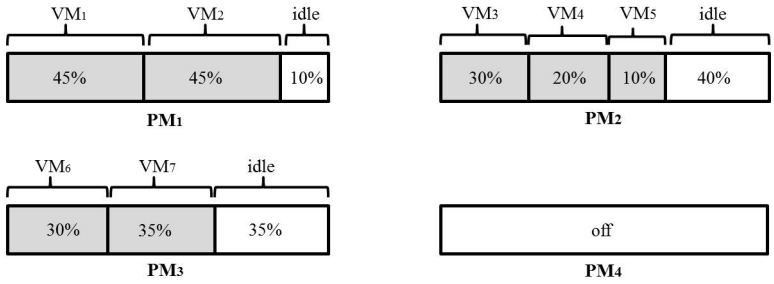


Figure 1.6: Initial Solution *i*

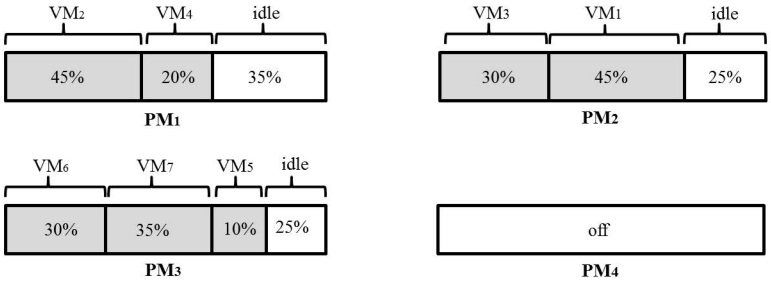


Figure 1.7: Target Solution s_1

Figure 1.6 shows the initial solution *i* for the dynamic VM placement problem. Seven VMs are placed over four PMs. PM_4 is in *off* state since no VM is running on it. Assuming that all the PMs have same amount of resources, the VMs should be dynamically distributed over different PMs in order to reduce energy consumption.

One such solution is shown in Figure 1.7. In order to reach solution s_1 from *i*, live migrations need to be performed. The list of migrations are : VM_5 from PM_2 to PM_3, migration of VM_1 from PM_1 to PM_2 and finally migrate VM_4 from PM_2 to PM_1.

Dynamic VM placement is explained in detail in chapter 2.

1.3 Literature Survey

Cloud computing is a very emerging topic and research is being conducted for energy efficient cloud environment. Virtual machine placement is one of the main research topics along with optimal resource allocation in cloud computing.

Urgaonkar et al. (41) in 2004 focuses on the application placement problem in a distributed systems environment. The aim is to maximize the number of applications which can be hosted on the distributed platform satisfying the resource constraints. Borst et al. (6), Ali et al. (1) and Amoura et al. (2) studied various resource management methods in the distributed systems environment (multi-server, multi-processor, grid).

Thiruvenkadam et al. (39) worked upon server overload which is one of the main problems in virtual machine placement. Comparison is shown between the greedy algorithm, round-robin algorithm and power save algorithm which are used for virtual machine placement. A new lively based scheduling algorithm is proposed which is quicker. Do et al. (11) focuses on application profiling for managing cloud resources efficiently. A Canonical Correlation Analysis (CCA) technique is presented which predicts application usage depending on their past usages. This helps in efficient VM placement.

Kantarci et al. (21) propose a Mixed Integer Linear Programming (MILP) method in order to place VMs on data centers (inter and intra placement) by virtualizing the background topology. The main objective is to minimize power consumption accounting CPU frequency, memory and bandwidth capabilities of the host machines. Dupont et al. (12) use Constraint Programming (CP) technique to create a flexible and energy-aware framework for VM placement in cloud federated DCs. The simulation proves that the framework is energy efficient with a low computation time.

Xu et al. (46) use genetic algorithm to solve virtual machine placement problem. It focuses on multi objectives, namely resource wastage minimization, power consumption and cost of thermal dissipation minimization. Simulation proves that the proposed approach solves the conflicting objectives while the bin packing algorithms cannot. Chaisiri et al. (8) propose an Optimal Virtual Machine Placement (OVMP) algorithm. The algorithm considers future demand and price uncertainty while solving using Stochastic Integer Programming (SIP) to minimize the cost of hosting VMs in a multiple cloud provider environment. The simulation results show the possibility of minimizing users' budgets using this algorithm.

Hyser et al. (18) address the dynamic VM placement problem which uses an existing

mapping for the initial point and then new placement solutions are generated for load balancing among hosts. Hermenier et al. (15) constructs the VM placement problem as a constraint satisfaction problem with objectives as minimization of number of used servers and migration costs. Khazaei et al. (23) proposed a technique for analysis based on an approximate Markov chain model using $M/G/m/m+r$ queueing systems for evaluating the performance of a cloud computation center. The author also published another work (22) which models Cloud Computing Centers using $M/G/m$ Queues.

In 2014, Xiao et al. (45) modeled the dynamic VM placement problem as an evolutionary game theoretic problem. It cites the problems in other centralized decision making algorithms which are being removed by the decentralized approach. One of the major problems in centralized intelligent algorithms like genetic algorithm, is the production of unreachable solutions from the existing mapping.

1.4 Research Motivation

Cloud computing is an emerging area which is helpful in providing energy efficient solution to distributed computing by the use of virtualization. Thus, virtual machine requests' distribution over the hosts form an integral part of the cloud computing architecture. The decision making process can be either centralized or decentralized, but the aim has to be to provide energy efficient solutions. The motivation for this work can be listed as follows.

- In order to provide energy efficient solutions, the objective is to use lesser number of PMs for processing the user requests. But the machines should not function at the maximum utilization. Thus, we need to find a solution which minimizes the number of PMs used with lesser utilization such that it is energy efficient.

- The centralized decision making process takes more time and is less energy efficient than the decentralized process. Thus game theoretic methods should be used in order to provide optimal solution to the VM placement problem, thus achieving Nash equilibrium.

- All the evolutionary algorithms like the genetic algorithm, simulated annealing algorithm do not provide the migrations needed to reach the optimal solution to the

dynamic VM placement problem, thus unreachable solutions are not taken care of (45).

1.5 Problem Statement

The dynamic virtual machine placement problem has been solved by many researchers. But they haven't considered the live migrations and thus may provide unreachable solutions to the problem. The primary objective of this research is to model a cloud computing framework such that the dynamic virtual placement problem can be solved while providing the live VM migrations to be executed in order to reach the solution.

This can be elaborated as follows :

- The motive is to model the dynamic placement problem in such a way that the states of virtual machines and physical machines are taken into account. Also the list of executable live VM migrations must be provided so that the optimal solution can be obtained.

- The problem if solved using centralized evolutionary techniques has a risk of providing unreachable solutions. Thus, decentralized approach using congestion game model must be used in order to provide the solutions to the dynamic virtual machine placement problem.

- In real world scenario, physical machines may act selfishly and do not cooperate to achieve an optimal solution to the problem. Thus both cooperative and non-cooperative approaches should be discussed in order to reach to an optimal solution.

The solution to the dynamic VM placement problem is to achieved taking into consideration the following constraints :

- *Capacity Constraint* - This condition ensures that the total resource requirements of all the VMs running on a specific PM should be less than or equal to the total resource availability of that particular PM.

- *Placement Constraint* - This constraint checks the criteria that a virtual machine should run on only a single physical machine.

- *SLA Constraint* - The cloud model to be incorporated in the real world should follow QoS parameters. Thus the service level agreement constraint keeps a tab on the

QoS parameters enlisted in the agreement and ensures that they are not violated when providing the optimal solution.

1.6 Research Contribution

The major contributions of this thesis can be written as:

- An energy consumption model is built to calculate the total energy consumption by physical machines, keeping in mind many factors. Then a decentralized approach is taken using cooperative game theory to optimally place virtual machines onto physical machines dynamically so as to minimize the energy consumption.
- During the decentralized decision making process, it may so happen that the physical machines act in a selfish manner and do not cooperate with each other to arrive at an optimal solution. Thus in this work, non-cooperative game theoretic approach is also proposed to provide optimal solution to the dynamic virtual machine placement problem.

1.7 Organization of Thesis

In this chapter, a summarized idea of cloud computing and virtual machine placement is given along with the literature survey. We even discuss the motivation and the problem statement. The remainder of the thesis has the following organization:

Chapter 2: A brief concept of dynamic VM placement problem and a literature survey of the existing research in dynamic VM placement problem are mentioned in this chapter.

Chapter 3: In this chapter, we have shown the solution to the dynamic VM placement problem in a cooperative game theoretic approach.

Chapter 4: This chapter provides a non-cooperative game theoretic approach in order to reach an optimal solution to the dynamic VM placement problem.

Chapter 5: We draw the conclusion of our research work and also give some points for the future work in this chapter.

CHAPTER 2

BACKGROUND AND PROBLEM FORMULATION

2.1 Introduction

VM placement problem can be distinguished into two kinds - static VM placement and dynamic VM placement. The cases during which static placement of VMs is considered are during system startup, or creation of new VMs which are to be placed onto PMs without any movement of existing VMs, or when VMs are shut down. Static placement is normally used during initial stages in offline mode, which may not be altered for prolonged time periods. But static VM placement does not consider the states of VMs and PMs; also the rate of arrival for user requests is not dealt with.

Dynamic placement of VMs modifies the mapping of VMs onto PMs dynamically at regular time intervals in order to provide optimal performance of the machines and not violating the Service Level Agreement (SLA). In this chapter, dynamic VM placement problem has been explained and then a detailed literature survey is given about the previous solutions to the VM placement problem. Also an energy consumption model is built which takes into consideration three sections of energy consumption, namely energy consumption of physical machines during different states, consumption of energy during state transitions and energy usage during live VM migrations.

2.2 Dynamic VM Placement in Cloud Computing

Dynamic placement of VMs places VMs based on an existing mapping which is in contrast to the static placement of VMs which starts with no mapping. Dynamic VM placement aims to reach optimal solutions from the existing mapping at minimum cost. This would not shut down or stop the already running VMs, thus the placement solution should

provide the list of live migrations to be executed in order to reach the optimal state from the existing state. The whole process is in contrast to the static placement of VMs where VMs can be stopped and restarted which increases the energy consumption and thus degrades the complete system performance. In order to execute dynamic placement of VMs, states of physical machines should also be considered.

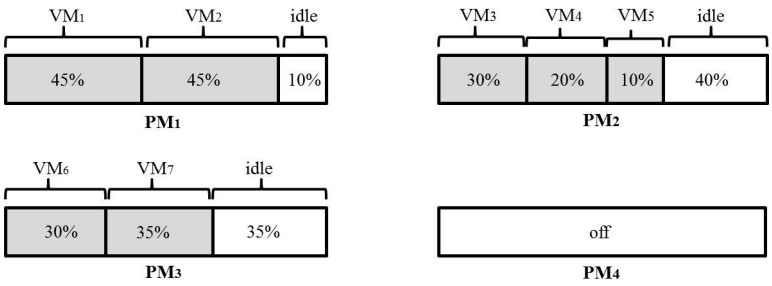


Figure 2.1: Initial Solution *i*

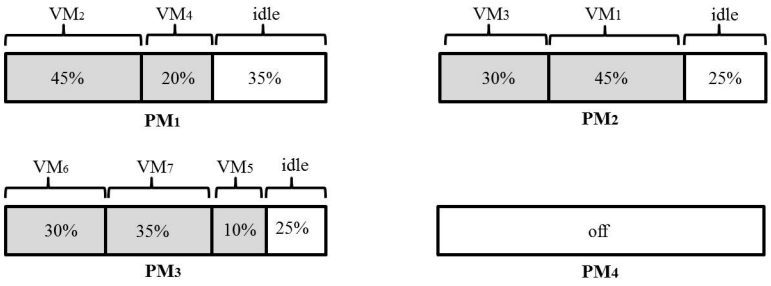


Figure 2.2: Target Solution s_1

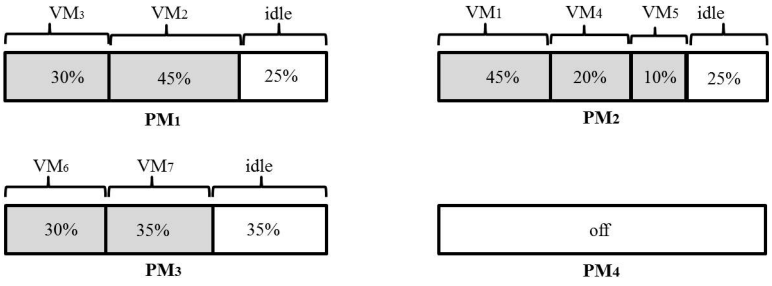


Figure 2.3: Target Solution s_2

Dynamic VM Placement problem can be illustrated in Figures 2.1, 2.2 and 2.3. Figure 2.1 shows the initial solution *i* for the dynamic VM placement problem. Distribution of seven VMs over four PMs is shown. PM_4 is in *off* state since no VM is running on it. Assuming that all the PMs have same amount of resources, the VMs should be dynamically distributed over different PMs in order to reduce energy consumption. One

such solution is shown in Figure 2.2. In order to reach solution s_1 from *i*, live migrations need to be performed. The list of migrations are : VM_5 from PM_2 to PM_3, migration of VM_1 from PM_1 to PM_2 and finally migrate VM_4 from PM_2 to PM_1. We can reach another solution s_2 from *i* as shown in Figure 2.3. Since there isn't much space in PM_1 and PM_2 for the direct interchange of VM_1 and VM_3, there needs to be an involvement of PM_3. Thus the migrations needed for solution s_2 to be reached are : the extra migration of VM_3 from PM_2 to PM_3, VM_1's migration from PM_1 to PM_2 and migrating VM_3 from PM_3 to PM_1. But in order to reach s_2 from i will result in more cost than to reach solution s_1 from initial placement i because of the migrations involving larger VMs. VM_1 and VM_3 could have been stopped and restarted in order to facilitate direct interchange but that would involve more cost. Also the involvement of PM_4 for the interchange of VM_1 and VM_3 would result in higher costs since it would then consider the cost of state change for PM_4.

2.3 Related Work

There is existing research concerned with finding solutions to the dynamic VM placement problem.

Hyser et al. (18) in 2007 studied the virtual machine placement scenario by developing an autonomic controller which dynamically maps the virtual machines onto the physical machines by following the users' policies. This work also differentiated the static VM placement and dynamic VM placement. The algorithms dealing with static VM placement start with a *clean slate*, that is no initial mapping. For example, loading goods from warehouse onto empty trucks. There isn't any need for these algorithms to consider the intermediate steps or the number of moves taken to reach the final state. This is in sharp contrast to the algorithms dealing with dynamic VM placement problem which must find optimal solutions from an initial mapping not a *clean slate*. In continuation with the same example as above, all the items are already loaded in the trucks. There isn't any warehouse. An optimal solution has to found with all executable migrations of items from one truck to another without violating any constraints.

In 2007, Wood et al. (43) presented a system called Sandpiper, which was was used for automatic monitoring and detection of system hotspots. It also provided a new mapping of VMs onto physical hosts and initiated the migrations too. Algorithms for hotspot

detection focuses on signaling a need for migration of VMs whenever SLA violations are detected either implicitly or explicitly. In the same year, Bobroff et al. (5) implemented an algorithm based on first-fit approximation to find solution to the problem of dynamic VM placement having the goal of price minimization. The problem was mapped as a bin packing one where the minimum number of PMs to be needed for the VMs was calculated and then a remapping was done for VMs onto PMs. The main disadvantage of the algorithm was that, it didn't look for unreachable solutions from the initial mapping.

A two phase process was developed by Hermenier et al. (16) in 2009 in order to find solutions to the dynamic VM placement problem. The consolidation manager named Entropy found solutions in two phases. The first phase finds out a placement keeping in mind the constraints, VM set and the CPU requirements. It also provides the likable configuration plan to achieve the desired mapping. The second phase tries to improve the result computed in the first phase. It takes into account a refined set of constraints and tries to minimize the number of migrations required. However very simple parameters are considered in this work and also server consolidation is not taken as a factor.

Liao et al. (28) in 2012 affirmed that dynamic VM placement problem faced three major challenges; multi-dimensional constraints, the initial state and the intermediary steps. They proposed a system called GreenMap which was a VM-based management framework to be able to execute live VM migrations considering the resource consumption of servers and energy consumption. Simulated annealing based heuristic is used for the optimization problem under the constraint of multi-dimensional resource consumption. The two objectives namely, reduction in energy consumption and performance degradation are balanced to give the desired output. But the GreenMap system doesn't address heterogeneous physical hosts which invariably form the real life server clusters.

2.4 Problem Formulation

There are P Physical Machines (PMs). Every PM has a state associated with it. The states included are : 1 = off, 2 = idle, 3 = ready and 4 = running. Thus we use an array PMS (1 X P) to specify the current state of a PM. For every PM, there are R different resources. Thus we use an array PMR (R X P) to specify the amount of resources that every PM has. Thus PMR_{ij} specifies the amount of $resource_i$ that PM_j has.

If there are V number of Virtual Machines (VMs), then an array VMR (R X V) is

created in order to forecast requirement of resources for VMs for a certain period of time. Thus VMR_{ij} indicates the requirement of i^{th} resource by the j^{th} VM. For all the V VMs, we require I number of performance indicators. In order to represent the performance requirements, we create an array VMP (I X V), where VMP_{ij} is the requirement of the i^{th} performance indicator for the j^{th} VM.

The result of the virtual machine placement problem is stored in a mapping array M (P X V). M_{ij} can hold values 0 or 1. The value 1 indicates that j^{th} VM is placed at the i^{th} PM, while the value 0 shows that j^{th} VM is not running on the i^{th} PM. We also require an array MP (I X V) which indicates the level of performance of all the VMs. MP_{ij} means the j^{th} VM is performing at the i^{th} level for a particular solution to the VM placement problem.

The VM placement problem aims to optimize the targeted features of the system by satisfying all the constraints. Performance optimization and cost optimization are the essential objectives of the dynamic VM placement problem. Performance optimization comes at an increased cost; thus most users do not aim for optimal performance as long as Service Level Agreement (SLA) is met. The SLA can be modeled as constraints which reduces the dynamic VM placement problem as a single objective problem focusing on cost optimization. Since reduction in energy consumption is the main part of the cost, we use it as the optimization objective.

The VM dynamic placement problem based on energy consumption (EC) (VDPPEC) can be formulated as follows:

$$VDPPEC = \min EC(M) \tag{2.4.1}$$

such that,

$$\begin{aligned} &C_{ri} \leq PMR_{ri} \\ &(C_{ri} = VMR \times M';\ r = 1, 2, \ldots., R;\ i = 1, 2, \ldots., P) \end{aligned} \tag{2.4.2}$$

$$\sum_{i=1}^{P} M_{ij} = 1 \qquad (j = 1, 2, \ldots.., V) \tag{2.4.3}$$

$$MP_{iv} \geq VMP_{iv} \ \ (i = 1, 2, \ldots., I;\ v = 1, 2, \ldots.., V) \tag{2.4.4}$$

Equation 2.4.1 shows the optimization objective of the VM dynamic placement problem, i.e. minimization of energy consumption (EC) for a particular placement solution (M).

The solution is arrived upon satisfying all the three constraints as shown in equations 2.4.2, 2.4.3 and 2.4.4. Equation 2.4.2 is the *capacity constraint* which checks the condition where the total resource requirements of the VMs running on a particular PM should be lesser than or equal to the total available resources of the specific PM. M' is the transposition of the mapping matrix (M). *Placement constraint* is represented in Equation 2.4.3 which states that a VM should run on only one PM. Equation 2.4.4 shows that the SLA should be met in VM placement solution.

2.4.1 Energy Consumption Model

The major part of energy consumption in a data center is produced by PMs which are running. There are other causes of energy consumption such as cooling apparatus but they form a much lesser percentage. Here we focus on the energy consumption or PMs which is divided into three sections.

Energy consumption in different states

Existing research indicates that energy consumption can be minimized by the adjustment of the states of PMs. Here 4 states of a PM are considered.

- *off* - This state consumes no energy.
- *ready* - PM is on but there are no active VMs on it. This state wastes energy consumption if PM is kept in this state for a long time. But if PM is switched off, much energy and time will be wasted to turn the PM back on to ready state.
- *idle* - This state exists as a result of a trade-off between off and ready states. PMs in this state consumes lesser energy than PMs in ready state. Also lesser time and energy will be required to turn a PM on from idle state to ready state.
- *running* - This state consumes the maximum energy.

The states and the transitions allowed are shown in Figure 2.4.

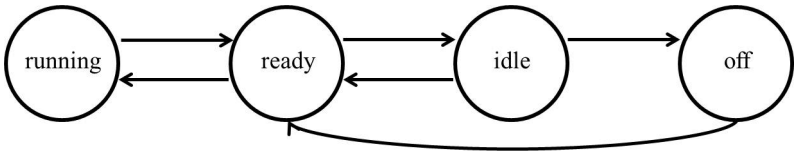


Figure 2.4: States and their transitions

It is observed that energy consumption varies according to load changes in a PM (46). Thus energy consumption of a PM depends on the PM's utilization. Thus the energy consumption for a PM can be modeled as:

$$EC(t) = FE \times t + x \times L(t)^y \tag{2.4.5}$$

In equation 2.4.5, $EC(t)$ represents the energy consumed in time period *t*, FE is the energy consumption that is fixed per unit time, $L(t)$ is the load of PM in time t, and x and y are the energy adjusting coefficients.

The above equation can be modified for modeling energy consumption of the i^{th} PM in a particular state, since the fixed energy consumption of a PM in every state is different.

$$EC_i(t) = FE_i(PMS_i) \times t + x_i \times L_i(t)^{y_i} \tag{2.4.6}$$

Equation 2.4.6 calculates the energy consumption of PM_i in time period t. $FE_i(PMS_i)$ is the fixed energy consumption per unit time of PM_i in state PMS_i. $L_i(t)$ is the load of PM_i for time period t, while x_i is the energy coefficient of the i^{th} PM and y_i is the relationship between load and energy of PM_i.

Energy consumption during state switch

Following rules are maintained while calculating energy consumption during switching of states of a PM.

- Energy consumption during switching between running and ready states can be ignored.
- State switching between off and ready states is slower than switches between idle and ready states.
- Switching off a PM costs lesser than keeping the PM in idle state.

Thus we use an array ECS (P X 4) to store the energy consumption of state switches since it is fixed for a particular state transition for a specific PM. The energy consumption during state switching of PM_i according to the array ECS can be seen in Figure 2.5.

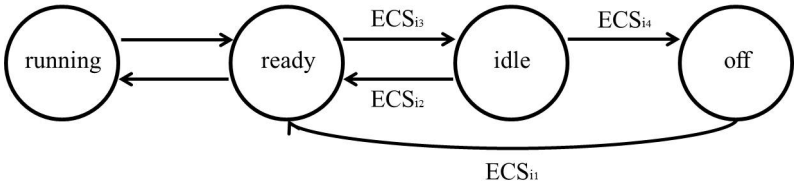


Figure 2.5: Energy consumption of state switching

Energy consumption during live VM migrations

The consumption of energy during live virtual machine migrations are of three parts;

- Energy consumption by the source PM for preparation for migration.
- Energy consumption by the target PM for reception and rebuilding of the migrated VM.
- Energy consumption by other physical equipment during VM migration.

We maintain three arrays in order for energy consumption during live VM migration. EVS (P X V) denotes the energy consumption by source PM during migration. EVS_{ij} is the energy consumed by PM_i for moving VM_j out.

EVT (P X V) denotes the energy consumption by target PMs for receiving the migrated VMs. Thus, EVT_{ij} is the energy consumed by PM_i for receiving the migrated VM_j.

EVX (P X P X V) is a three dimensional array which denotes the energy consumption during migration. Thus EVX_{ijk} is the energy consumed during migration of VM_k from PM_i to PM_j.

Thus the energy consumption during live VM migrations can be modeled as follows:

$$EVS_{ij} = \alpha_i \times SZ_j + \beta_i \tag{2.4.7}$$

$$EVT_{kj} = \alpha_k \times SZ_j + \beta_k \tag{2.4.8}$$

$$EVX_{kj} = \gamma_{ik} \times SZ_j \tag{2.4.9}$$

where, α_i, β_i, α_k, β_k and γ_{ik} are energy adjusting coefficients and SZ_j is the size of VM_j.

The overall energy consumption model

The energy consumption due to the switching of states during the entire migration process can be simplified. On comparison of the array PMS before and after the whole migration process, an array $PMST$ (4 X P) can be derived, which represents the state switching of all PMs. Thus,

- $PMST_{1j}$ = 1 means PM_j need state transition from off to ready.
- $PMST_{2j}$ = 1 means PM_j need state transition from idle to ready.
- $PMST_{3j}$ = 1 means PM_j need state transition from ready to idle.
- $PMST_{4j}$ = 1 means PM_j need state transition from idle to off.

Thus the total energy consumption by state switching of PMs (EPS) for a particular solution (M) of the VM placement problem can be calculated as:

$$EPS(M) = \sum_{i=1}^{P} \sum_{k=1}^{4} (ECS_{ik} \times PMST_{ki}) \tag{2.4.10}$$

In order to calculate the total energy consumption during live migration, we create 3 arrays by comparing the array M before and after migration. Array $VMLM$ (P X P X V) is created to show the live migrations for all the VMs. Thus, $VMLM_{ikj} = 1$ means that $VM_j's$ migration from PM_i to PM_k is needed. Similarly 2 other arrays VMS (P X V) and VMT (P X V) are created to show the sources of migrations and their targets respectively. Thus $VMS_{ij} = 1$ means that VM_j is migrated from PM_i and $VMT_{kj} = 1$ means that VM_j is migrated to PM_k.

Thus the total energy consumption during migration of VMs (EVM) for a particular solution (M) can be calculated as:

$$\begin{aligned} EVM(M) = & \sum_{i=1}^{P} \sum_{j=1}^{V} (EVS_{ij} \times VMS_{ij}) + \sum_{k=1}^{P} \sum_{j=1}^{V} (EVT_{kj} \times VMT_{kj}) \\ & + \sum_{i=1}^{P} \sum_{k=1}^{P} \sum_{j=1}^{V} (EVX_{ikj} \times VMLM_{ikj}) \end{aligned} \tag{2.4.11}$$

The total energy consumption by all the PMs in different states (EPM) for a particular solution (M) can be given as:

$$EPM(M) = \sum_{i=1}^{P} EC_i(t) \tag{2.4.12}$$

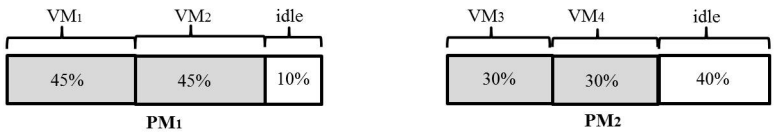
Thus the total energy consumption by the VM placement problem (TEC) from initial solution M' to solution M can be denoted as:

$$TEC(M) = EPS(M) + EVM(M) + EPM(M) - EPM(M') \tag{2.4.13}$$

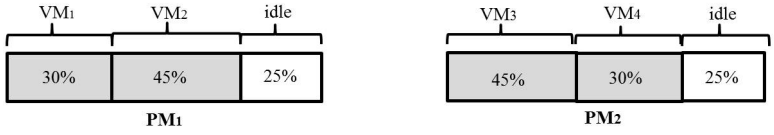
where, $EPM(M')$ is the energy consumption of all PMs under initial solution M'.

2.4.2 Solution to Dynamic VM Placement Problem

Dynamic virtual machine placement problem is NP-hard. There has been existing research where attempts have been made to map the dynamic VM placement problem to bin-packing problem which led to the development of evolutionary algorithms like, genetic algorithms or particle swarm optimization algorithm. As explained in Section 2.2, many problems arise when VMs have to be placed dynamically. While the evolutionary algorithms like genetic algorithms tries to provide optimal solutions to the problem, there may be solutions which are unreachable.



(a) Initial solution sol_0



(a) Optimal solution *sol*

Figure 2.6: Unreachable Solution

Figure 2.6 shows an example of unreachable solution to the dynamic VM placement problem. Figure 2.6(a) shows the placement of 4 VMs onto 2 PMs which form the initial

solution sol_0. The optimal solution *sol* is shown in Figure 2.6(b) which brings down the energy consumption by a considerable amount. But the solution is unreachable since there is no executable migration route from *(a)* to *(b)*. In order to achieve the solution, a third PM has to be switched on to handle the migrations which will again consume more energy. Thus, in case of evolutionary algorithms, judgment has to be made whether the optimal solution is reachable or not.

In this thesis, decentralized decision making is used with the help of congestion game theory in order to find solutions to the dynamic VM placement problem. In *Congestion Games*, a group of players is modeled to share a resource set. Every player selects a subset of resources from the resource set in order to maximize the payoff (32). Here, PMs will be modeled as players which will select a subset of VMs from the VM set in order to minimize the consumption of energy. Nash Equilibrium is attained after a finite number of iterations. This intelligent algorithm uses the initial mapping to obtain the optimum solution and also generates the list of executable VM live migrations to reach the optimal state. Thus no solution is unreachable.

2.5 Conclusion

In this chapter, the dynamic virtual machine placement problem has been discussed in detail. Also, the already existing work concerned with finding solutions to the dynamic VM placement problem is discussed. The problem statement is formulated along with the discussion that the problem will be solved in this thesis in a decentralized manner with the help of congestion game theory. The next chapter focuses on finding solutions to the dynamic VM placement problem in cooperative game theoretic manner.

CHAPTER 3

COOPERATIVE GAME THEORY FOR DYNAMIC VM PLACEMENT

3.1 Introduction

Minimization of power consumption in data centers has become one of the major problems in recent times. It has been observed that one of the most effective ways to save power consumption is to optimize the VM placement dynamically in a virtualized data center.

Decentralized approach is used for the dynamic placement of VMs over PMs in order to optimize the energy consumption. A novel algorithm based on congestion game theory has been proposed to solve the VM placement problem. It is proved that the algorithm reaches Nash Equilibrium in polynomial time. Also the algorithm can take an initial mapping as input and produce a list of live VM migrations as an output. It is considered in this chapter that all the physical machines cooperate with one another to achieve the optimal solution; thus *cooperative game* is used among physical machines.

3.2 Game Theory

Turocy et al. (40) defines game theory as the formal study of cooperation and conflict. Hotz (17) describes a game as a set of players and their possibilities to play the game by following some rules (strategies). The players can be individuals, agents or organizations. The main subject of game theory are the situations where the result is mattered not only by the decision of a single player but others as well.

The earliest instance of formal game theoretic analysis is Antoine Cournot's study of duopoly in 1838. In 1921, a formal theory of games was suggested by Emile Borel. This was further studied upon by John von Neumann's *theory of parlor games* in 1928. The basic terminology and setup of game theory was established in *Theory of Games and Economic Behavior* by von Neumann and Oskar Morgenstern in 1944. John Nash in 1950 showed that the games with finite number of players always have a point of equilibrium where the players choose their best strategy taking into consideration the strategies of other players. This equilibrium is known as *Nash Equilibrium* named after John Nash. This pivotal point in non cooperative game theory has been used in various fields from sociology, biology to computer science.

3.2.1 Types of Game Scenarios

There are mainly two types of game scenarios:

- Cooperative Game
- Non-Cooperative Game

Cooperative Game

Xhafa et al. (44) defines cooperative game as the game scenario where players form coalitions in advance to discuss their actions. Cooperative game theory investigates such coalition games by studying how successfully a coalition divides its proceeds (40).

The applications of cooperative game theory is mainly seen in cases related to political science or international relations. In cloud computing, cooperative game theory is used when data centers are managed by a single service provider or the providers form a coalition such that the strategies of virtual machine placement are known by all the physical machines and respective actions are taken to maximize the payoffs of all the SPs.

Non-Cooperative Game

The term *non-cooperative* implies that this type of game models the process of players who are making choices thinking about their own interest.

The non cooperative game theory relates to realistic cloud computing where the service providers do not form coalitions and a decision is made considering the selfish actions of individual providers who want to maximize their profit.

3.2.2 Games in Normal Form

The representation of a game in *normal form* or *strategic form* is (19):

- The set of players $N = \{1 \ldots . n\}$.
- Player i has a set of actions a_i which are normally referred to as *pure strategies*.
- The set of all pure strategies is denoted by $a = (a_1 \ldots . a_n)$.
- Player *i* has a payoff represented as the function of the action vectors is denoted by $u_i : A \rightarrow IR$ where $u_i(a)$ is *i's* payoff if *a* is the strategy taken.

Normal form games are often represented in the form of a table. The most common example is *prisoner's dilemma* represented in Figure 3.1.

		Player 2	
		C	D
Player 1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

Figure 3.1: A prisoner's Dilemma Game

In prisoner's dilemma, the number of players is two with each having two pure strategies, where $a_i = \{C, D\}$; C is for *cooperate* and D is for *defect*. C indicates the payoff to the row player (player 1) as a function of the pair of actions, while D is the payoff to the column player (player 2). The game is explained as follows. Both the players are caught committing a crime and are investigated in different cells in a police station. The prosecutor comes to each of them and tells each of them:
If you provide a confession and accept to testify against the other; and if the partner doesn't confess; you will be set free. If both of you accept committing the crime, you will both be sent to jail for two years. If you do not accept committing the crime but your partner does, you will be given a sentence of three years imprisonment. If none of you

confess; only one year punishment will be given due to the lack of evidence.
So the payoff matrix shows the imprisonment time in years. The term cooperate means that the partners cooperate with each other. The term defect means that you are accepting the crime and agreeing to testify, and so you are breaking the agreement which you two have.

3.2.3 Nash Equilibrium

If a set of strategies for the players constitute a Nash Equilibrium, it means that none of the players can benefit by altering his/her strategy unilaterally.

A strategy a_i is a best reply, also known as a best response, of player i to a set of strategies $a_{-i} \in a_{-i}$ for the other players if

$$\mathrm{u_i}(\mathrm{a_i}, \mathrm{a_{-i}}) \geq \mathrm{u_i}(\mathrm{a'_i}, \mathrm{a_{-i}})\,; \qquad \forall\ \mathrm{a'_i} \tag{3.2.1}$$

A profile of strategies $a \in A$ is a pure strategy Nash equilibrium if a_i is a best reply to a_i for each i. That is, a is a Nash equilibrium if

$$\mathrm{u_i}(\mathrm{a_i}, \mathrm{a_{-i}}) \geq \mathrm{u_i}(\mathrm{a'_i}, \mathrm{a_{-i}})\,; \qquad \forall\ \mathrm{i}, \mathrm{a'_i} \tag{3.2.2}$$

A pure strategy Nash equilibrium only states that the action taken by each agent be the best against the actual equilibrium actions taken by the other players, and not necessarily against all possible actions of the other players.

In *prisoner's dilemma*, Nash Equilibrium occurs if both player 1 and 2 cooperate (C). If any of the players change his/her strategy, no player can benefit.

3.3 Literature Survey for Cooperative Game Theory

The process flow in centralized cloud computing is shown in Figure 3.2. Users on accessing services provided by the cloud service providers, send requests to the *request manager*. The request manager has information of the details of every request. These details are given to the *data center controller* which has information of all the DCs provided

by the individual *DC managers*. The DC controller takes a centralized decision with respect to both the information (request details and data center details) and then sends the request to a particular data center manager. *Data center manager* thus has the details of the request queue sent to that particular data center by data center controller. It fetches the information of all the PMs present in the data center from the *Hypervisor (Virtual Machine Monitor)*. A centralized decision is taken by the data center manager and the requests are sent to hypervisors which process the requests in virtual machines thus implementing virtualization. The request is processed and response is given back to the user by the *response manager*. This explains the two tier central decision making in cloud computing.

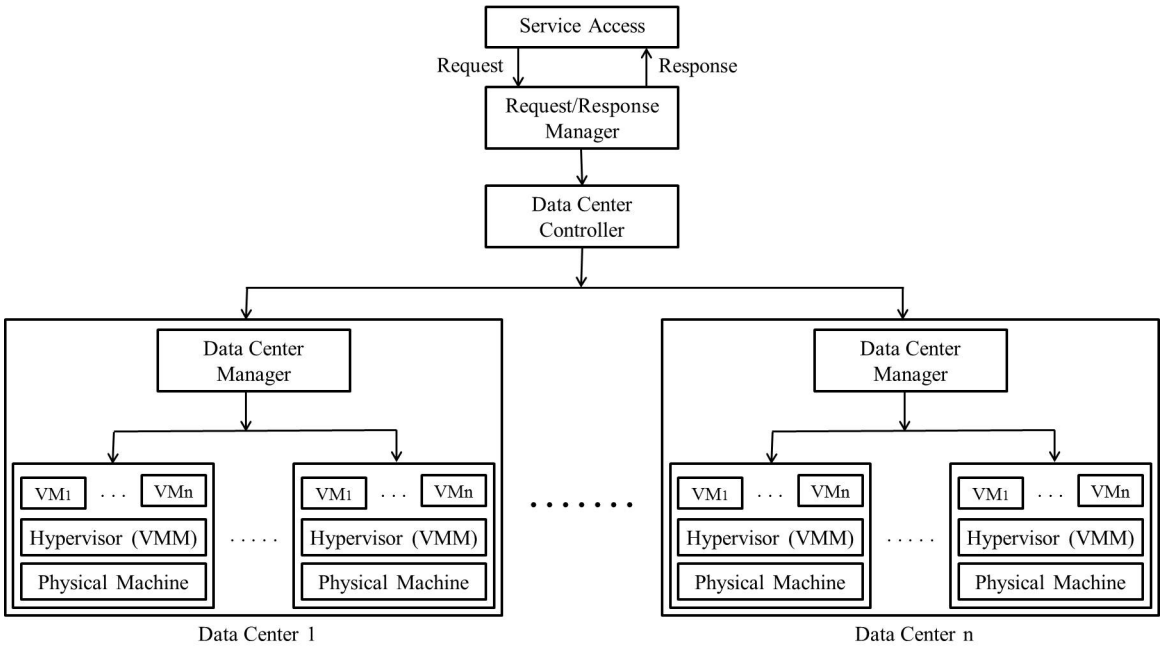


Figure 3.2: Centralized Decision Making in Cloud Computing

In decentralized decision making, the two tier decision making process is taken by the players. In the first tier, decision to distribute requests over data centers are taken by all the data center managers which act as players. Next the players are the physical machine which take the decision to optimally place virtual machines. The players may or may not cooperate with each other in order to make a decision. In this thesis, the decentralized decision making is implemented using game theory. Various works have already been done in the field of cooperative game theory in cloud computing.

Wei et al. (42) uses cooperative game theory in order to find solution of the QOS constrained problem of allocation of resources. Here problem solution is done in 2 steps. First optimal problem solution is done by every participant independently without re-

source multiplexing. Next a mechanism is designed by considering strategies which are multiplexed, of all the players. Existence of Nash Equilibrium is shown if resource allocation problem has feasible solutions. This paper creates a useful analytical tool for the solution of optimal scheduling problem.

No consideration for multi-tier architecture of web services and lack of emphasis on stable placement of applications are the drawbacks in (42). These disadvantages are removed by Lee et al. (27) where an evolutionary approach of game theory is designed for stable and adaptive placement of applications. This paper implements Nuage which uses an evolutionary game theoretic approach in order for stable adaptive deployment of applications. The main goal of this work is to N applications on M hosts with the goal that applications adapt their locations and allocation of resources is done on the basis of resource and workload availability. In this work only CPU time share is considered for assignment of resources to each VM. Resources such as memory space and network bandwidth should also be considered for allocation of resources. Also there isn't any comparison of the evolutionary game theory with existing optimization algorithms.

General colocation game and process colocation game were introduced by Londoño et al. in (29) in order to distribute resources to infrastructure providers. This work considers a cooperative game theoretic framework where the objectives for resource management were to maximize resource utilization and minimize total cost of the allocated resources. The work also proves that achieving nash equilibrium is NP-complete. The main drawbacks of this this work are though it provides best response computation, it is an expensive framework. Also it does not consider P2P systems to optimize resource management without the need for central authority.

Cooperative VM management for multi-organization environment in cloud computing was done by Niyato et al. (34). Three types of resources are considered for VM management; private cloud, on-demand plans and reservation of public CSP. The algorithm works in two steps. First an optimization model is formulated for cooperative organizations and optimal VM allocation problem is solved in order to minimize the total cost. Then network game is used to analyze the cooperative framework. This work doesn't consider the stochastic nature of demand.

Resource allocation in Horizontal Dynamic Cloud Federation Platform (HDCF) environment is studied by Hassan et al. (14). The work presents a cost effective and scalable solution to the resource allocation problem using game theory. It studies both cooperative and non-cooperative resource allocation games as well as both decentralized and central-

ized algorithms are presented in order to find optimal solutions. The major drawback in this framework is that it doesn't take into consideration the dynamic nature of clouds where hundreds of clouds leave and join the federation in a dynamic manner.

Mao et al. (31) worked upon the problem of cloud service deployment by modeling it as a congestion game. Only cost and quality of resources were considered. Every service acts as a player which chooses a subset of resources for the maximization of his payoff. It is shown that Nash equilibrium is reached in polynomial time. This work can be further implemented in seeking solution to the placement of multiple cooperative heterogeneous components over many cloud services.

New resource allocation game models (CT-RAG, CS-RAG) were introduced for problem solving in cloud computing by Sun et al. (37). Existence of Nash equilibrium is shown but the work considers static game with no representation of fairness of tasks. The problem of allocation of resources for PMs in cloud computing based on the uncertainty principle of game theory and coalition formulation was studied by Pillai et al. (35). It is shown that the solution gives higher request satisfaction and better resource utilization.

3.4 Observations

Table 3.1: Cooperative Game Theory in Distributed Resource Management

Researcher	Game	Environment	Work
2009			
Wei et al. (42)	Cooperative	Cloud	Solves QOS constraint resource problem by two step method and shows the existence of Nash equilibrium for an allocation game feasible solutions.
Londono et al. (29)	Cooperative	Cloud	Introduces both General Colocation Game (GCG) and Process Colocation Game (PCG) for the solution to resource management problem in cloud computing.

2010			
Lee et al. (27)	Cooperative	Cloud	Game theoretic approach to stable and adaptive application placement in cloud environment considering multi-tier architecture. The work considers only CPU time share for application deployment.
Xhafa et al. (44)		Grid	Presents a survey of the game theoretic models used for allocation of resources in grid systems and their solution using meta heuristic models.
2011			
Niyato et al. (34)	Cooperative	Cloud	Formulates model for optimal virtual machine allocation in cooperative organizations to minimize the total cost. The stochastic nature of demand is not considered.
Niyato et al. (33)	Cooperative	Cloud	Studies the cooperative behaviors of multiple cloud providers to present a hierarchical model of cooperative game. Solution is obtained using stochastic linear programming game model.
Hassan et al. (14)	Cooperative and Non-cooperative	Cloud	Studies both cooperative and non-cooperative games for the allocation problem of resources in HDCF platform presenting both centralized and decentralized algorithms.
2012			
Ge et al. (13)		Mobile Cloud	Formulates the congestion game model in order to find the solution to the energy minimization problem in mobile cloud computing. Existence of Nash Equilibrium is proved.

Lu et al. (30)	Cooperative	Cloud	Cooperative sharing of the resource and revenue in a cloud federation environment is considered from the game theoretic perspective. Cloud providers form coalition in order to increase profit and resource utilization.
2013			
Mao et al. (31)	Cooperative	Cloud	The cloud service deployment problem is modeled as a congestion game with price and quality as the performance objectives. Each service is treated as a player which selects a subset of resources to maximize payoff.
2014			
Sun et al. (37)	Cooperative	Cloud	Group participation game strategy is proposed for resource allocation problem. Two game models namely CT-RAG and CS-Rag are discussed. This work considers only static game with CPU as the sole resource.
Pillai et al. (35)	Cooperative	Cloud	Solution to resource allocation problem is obtained based on formation of coalition among agents and the uncertainty principle of game theory. This results in better utilization of resources.

3.5 System Model

With increasing dependence on Cloud Service Providers (CSP) by consumers for computing needs, there has been an enhanced requirement of a specific level of QoS which has to be followed by the CSPs. CSPs aim to meet the QoS parameters which are specified in the negotiated Service Level Agreement (SLA). Thus, an improved cloud architecture

has to be developed which will cater to the needs of the consumers as well as the CSPs. The problem model that has been used in this chapter is depicted in Figure 3.3.

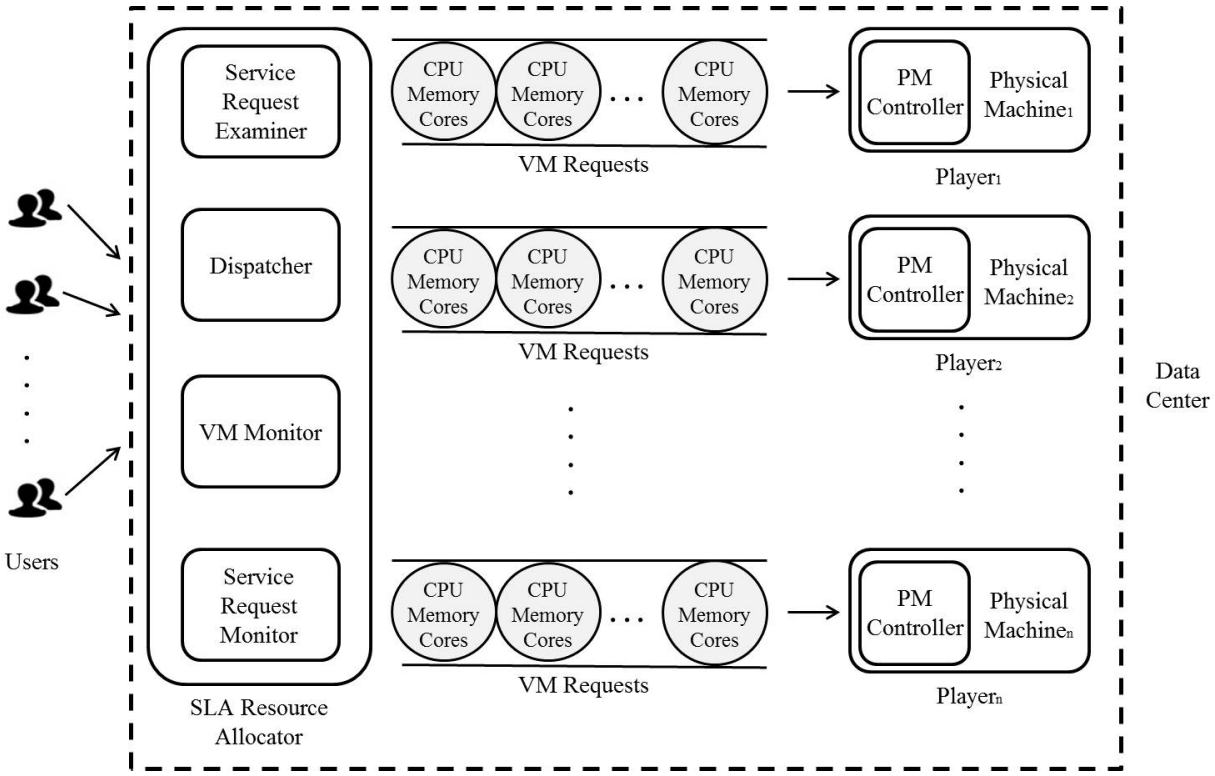


Figure 3.3: Proposed System Model

The cloud architecture being referred here has many entities. *Users* or brokers which act on the behalf of users, submit their requests to the cloud data center. *SLA Resource Allocator* provides the platform between the CSP/DC and the user/broker. On submission of a user request, the *Service Request Examiner* checks the request for QoS requirements' parameters to decide whether or not to accept the user request. It makes sure that no overloading occurs by keeping a tab on the availability of resources obtained from the *VM Monitor* and already available system resource usage from the *Service Request Monitor*. Keeping these in mind, the examiner assigns the user requests to VMs and decide the requirements of resources for the allotted VMs.

VM requirements are given in the form of a vector $x.(\overrightarrow{cpu}) + y.(\overrightarrow{mem}) + z.(\overrightarrow{cores})$, where $\overrightarrow{cpu}$ is the CPU requirement in GHz, $\overrightarrow{mem}$ is the memory requirement in GB and $\overrightarrow{cores}$ is the requirement for the number of cores. x, y and z are the multiplying factors constraint to $x + y + z = 1$

The *VM Monitor* keeps a check on the VMs' availability and resource requirements. *Dispatcher* begins executing the accepted user requests on the allotted VMs. *Service Request Monitor* keeps a check on the progress of the execution of the service requests.

All Physical Machines (PMs) have a *PM Controller* which accepts the VM Request queue sent from the dispatcher. The PM Controllers are as players for the cooperative game theory which take the decision on the dynamic VM placement problem in order to minimize energy consumption.

3.6 Problem Statement

Figure 3.3 refers to the system model which is used in order to find the solution to dynamic VM placement problem.

There are n Physical Machines (PMs). Every PM has a state associated with it. The states included are : 1 = off, 2 = idle, 3 = ready and 4 = running. Thus we use an array PMS (1 X P) to specify the current state of a PM. For every PM, there are 3 resources, namely CPU (GHz), Memory (GB) and number of cores. Thus we use an array PMR (3 X n) to specify the amount of resources that every PM has. Thus PMR_{ij} specifies the amount of $resource_i$ that PM_j has. The dynamic VM placement problem in this chapter is solved by *Cooperative Game Theory* where the players are the physical machines. Thus the resource status information is shared by all the PMs which aids in the problem solving process.

There are V_i number of VMs in PM_i. Let the maximum number of VMs present in one particular PM be $maxV$.

$$\max V \; = \; \max(V_i) \quad ; \; 1 \leq i \leq n \tag{3.6.1}$$

VM requirements are given in the form of a vector $x.(\overrightarrow{cpu}) \; + \; y.(\overrightarrow{mem}) \; + \; z.(\overrightarrow{cores})$, where $\overrightarrow{cpu}$ is the CPU requirement in GHz, $\overrightarrow{mem}$ is the memory requirement in GB and $\overrightarrow{cores}$ is the requirement for the number of cores. x, y and z are the multiplying factors constraint to $x + y + z = 1$.

Thus a 3-dimensional array VMR (3 X maxV X n) is created in order to forecast requirement of resources for VMs for a certain period of time. Thus VMR_{ijk} indicates the requirement of i^{th} resource by the j^{th} VM residing in k^{th} PM. For all the V VMs where $V \; = \; \sum_{i=1}^{n} V_i$, we require I number of performance indicators. In order to represent the performance requirements, we create an array VMP (I X V), where VMP_{ij} is the requirement of the i^{th} performance indicator for the j^{th} VM. This performance indicator

acts as the SLA indicator.

The result of the virtual machine placement problem is stored in a mapping array M (P X V). M_{ij} can hold values 0 or 1. The value 1 indicates that j^{th} VM is placed at the i^{th} PM, while the value 0 shows that j^{th} VM is not running on the i^{th} PM. We also require an array MP (I X V) which indicates the level of performance of all the VMs. MP_{ij} means the j^{th} VM is performing at the i^{th} level for a particular solution to the VM placement problem.

The VM dynamic placement problem based on energy consumption (EC) (VDPPEC) can be formulated as follows:

$$VDPPEC = \min TEC(M) \tag{3.6.2}$$

where,

$$TEC(M) = EPS(M) + EVM(M) + EPM(M) \tag{3.6.3}$$

such that,

$$\begin{aligned} &C_{ri} \leq PMR_{ri} \\ &(C_{ri} = VMR \times M'; r = 1, 2, \ldots., R; i = 1, 2, \ldots., P) \end{aligned} \tag{3.6.4}$$

$$\sum_{i=1}^{P} M_{ij} = 1 \qquad (j = 1, 2, \ldots., V) \tag{3.6.5}$$

$$MP_{iv} \geq VMP_{iv} \;\; (i = 1, 2, \ldots., I; \; v = 1, 2, \ldots.., V) \tag{3.6.6}$$

Equation 3.6.2 shows the optimization objective of the VM dynamic placement problem, i.e. minimization of energy consumption (EC) for a particular placement solution (M).

The solution is arrived upon satisfying all the three constraints as shown in equations 3.6.4, 3.6.5 and 3.6.6. Equation 3.6.4 is the *capacity constraint* which checks the condition where the total resource requirements of the VMs running on a particular PM should be less than or equal to the total available resources of the specific PM. M' is the transposition of the mapping matrix (M). *Placement constraint* is represented in Equation 3.6.5 which states that a VM should run on only one PM. Equation 3.6.6 shows that the SLA should be met in VM placement solution.

3.7 Optimization Algorithm for VDPPEC

In this section, the algorithm based on congestion game theory for cooperative physical machines has been discussed which is used for solving the dynamic VM placement problem. In order for better convenience for description of the algorithm, the symbols along with their meanings have been listed in Table 3.2.

The algorithm is executed dynamically whenever any SLA constraint is violated. The status of all the physical machines is shared by all the players. Thus, it is cooperative game theory.

Table 3.2: List of symbols and their meanings

Symbols	**Meanings**
n	Number of physical machines
V	Number of virtual machines
ρ^n	Initial mapping of V VMs onto n PMs
δ^n	Optimal mapping of V VMs onto n PMs after dynamic VM placement
δ^{α}_{p-1}	A strategy α for PM_{p-1}
ϑ	3-dimensional matrix (n X n X V) showing the live migrations to be executed. Thus, $\vartheta_{ikj} = 1$ means that $VM'_j s$ migration from PM_i to PM_k is needed.
S	Strategy Matrix to store the collection of strategies for all the PMs. Thus S_{ij} means the j^{th} strategy for the i^{th} PM.
$payoff_i(\delta^{\alpha}_i)$	Total Energy Consumption if strategy α of PM_i is taken.

Algorithm 1 is the calling module. Is takes the initial mapping of V virtual machines onto n physical machines as input and produces the best strategy or the optimal mapping with minimum energy consumption as the output. Also the the migrations needed to reach to the optimal solution is given as output.

The algorithm uses the congestion game model which models a resource set being shared by a group of players. Here every player selects a subset of resources from the resource set in order to maximize his own payoff; here maximizing the payoff means minimizing the energy consumption. Here, the physical machines are treated as players which select a subset of VMs in order to minimize the total consumption of energy. Thus algorithm 1 calls algorithm 2 for each player in order to reach Equilibrium at every step.

Every player, that is each Physical machine acts as a player and attempts to achieve its own equilibrium by going through algorithm 2.

Algorithm 2 finds the best strategy for the i^{th} PM. It takes as input the best strategy

Algorithm 1: Equilibrium for n players

Input: n Physical Machines, V Virtual Machines
Initial mapping for n players; $\rho^n = (\rho_0^n, \rho_1^n, \ldots\ldots, \rho_{n-1}^n)$
Result: Best strategy for n players; $\delta^n = (\delta_0^n, \delta_1^n, \ldots\ldots, \delta_{n-1}^n)$
Executable live migrations matrix; ϑ
for $(i = 0; i < n; i++)$ **do**
  Nash_Equilibrium(i);

Algorithm 2: Nash_Equilibrium(p)

Input: Best strategy for (p-1) Physical Machines
; $\delta^{p-1} = (\delta_0^{p-1}, \delta_1^{p-1}, \ldots\ldots, \delta_{p-2}^{p-1})$
Output: Best strategy for p players; $\delta^p = (\delta_0^p, \delta_1^p, \ldots\ldots, \delta_{p-1}^p)$
$\exists \delta_{p-1}^{\alpha} \in S_{(p-1)j}$,
$payoff_{p-1}(\delta_{p-1}^{\alpha}) \leq payoff_{p-1}(\delta_{p-1})$; $\forall \delta_{p-1} \in S_{(p-1)j}$
$\delta_{p-1}^p \leftarrow \delta_{p-1}^{\alpha}$
$\delta_i^p \leftarrow \delta_i^{p-1}$, $i \in [0, p-2]$
$\delta^p = (\delta_0^p, \delta_1^p, \ldots\ldots, \delta_{p-2}^p, \delta_{p-1}^p)$
$j = 0$;
for $(i = 0; i < (p-1); i = j)$ **do**
  if $(payoff_i(\delta_i^p) \leq payoff_i(\delta_i)$; $\forall \delta_i \in S_{ij})$ **then**
    j++;
  else
    $\exists \delta_i^{\alpha} \in S_{ij}$,
    $payoff_i(\delta_i^{\alpha}) \leq payoff_i(\delta_i)$; $\forall \delta_i \in S_{ij}$
    $\delta_i^p \leftarrow \delta_i^{\alpha}$
    $j = 0$;

for (i-1) PMs and gives as output the best strategy of i PMs. First the i^{th} PM looks for the best strategy which will consume the least energy. Then all the $(i-1)$ PMs change their strategies accordingly so that equilibrium is maintained, that is optimal energy consumption takes place. This algorithm thus produces the best strategy for every step for all the n physical machines.

There are n players (Physical Machines) and V virtual machines. So in each step in order to achieve Nash Equilibrium for a player the time complexity will be $O(nV)$. The algorithm for finding out the equilibrium for a particular step has to be repeated for n players. Thus the time complexity to achieve Nash Equilibrium for n players will be $O(n^2V)$. Thus this algorithm obtains the Nash Equilibrium in cooperative congestion game for dynamic VM placement at a particular time frame in polynomial time.

3.8 Simulation and Results

In this section, simulation parameters are defined and experimental results are discussed to evaluate the performance of Algorithm 1 and 2, which aim to optimally provide solutions tot he dynamic VM placement problem. The simulation has been conducted using an in-house simulation using JAVA on a desktop computer with Intel (R) Core (TM) i7-3770 processor, 3.4 GHz and 4 GB memory.

3.8.1 Energy Consumption Model

The parameters' values for the energy consumption model which is proposed in this chapter are given below (45).

The values of x_i and y_i are determined via function fitting graph related to CPU utilization and are fixed at 61.06 and 2 respectively. Thus Equation 2.4.6 which calculates the consumption of energy by PMs in different states can be written as :

$$EC_i(t) \; = \; FE_i(PMS_i) \; \times \; t \; + \; 61.06 \; \times \; L_i(t)^2 \tag{3.8.1}$$

It is considered in the experiments that all the PMs are homogeneous in nature. Thus the parameters' values are same for all the PMs.

Table 3.3: Values for The Energy Consumption Parameters

States	**Running**	**Ready**	**Idle**	**Off**
Running	FE_i(running) = 32.2708	-	-	-
Ready	-	FE_i(ready) = 32.2708	ECS_{i3} = 3.5	-
Idle	-	ECS_{i2} = 7.5	FE_i(idle) = 0.5	ECS_{i4} = 5.5
Off	-	ECS_{i1} = 0	-	FE_i(off) = 0

Table 3.3 gives us the values for the parameter ECS and FE which are used to measure the PM's energy consumption of state switching.

To determine the energy consumption during live VM migration, the parameters have their values fixed; α_i = 0.256, β_i = 10.0825, α_k = 0.256, β_k = 10.0825 and γ_{ik} = 0.5.

3.8.2 Experimental Results

Six sets of experiments are performed and the results are compared with results of Best Fit algorithm. Best Fit Algorithm is a centralized algorithm which aims to place VMs into the PM with the least available space and with no SLA violations, such that the PMs are fully utilized. The steps involved in best fit procedure are defined in Algorithm 3.

Algorithm 3: Best Fit

Input: p Physical Machines, n Virtual Machines
Result: Best Fit placement strategy for n Virtual Machines
for $(i = 0; i < n; i = i + 1)$ **do**
  Sort p PMs in increasing order of free space (remaining capacity)
  for $(j = 0; j < p; j = j + 1)$ **do**
    if $(Requirement(VM_i) \leq Capacity(PM_j)$ **then**
      Place VM_i into PM_j
      break;

First three experiments are conducted by keeping the number of PMs fixed at 40. The number of VM requests are varied from 10 to 150 and results obtained from best fit algorithm (centralized approach) and cooperative game theoretic process (decentralized approach) are compared.

Figure 3.4 shows the results of number of PMs used against number of VMs, keeping the number of available PMs fixed at 40. It is seen that for lower number of VMs, the number of PMs used is similar for both best fit and cooperative game theoretic approach. But as the number of VMs increases, there is a marked difference between the results of two approaches. Cooperative game theoretic algorithm optimizes the placement strategy and uses lesser number of PMs for the placement of VMs onto PMs. For large number of VMs, the best fit strategy uses all the PMs for the placement.

The result for energy consumption vs no. of VMs for a fixed number of available VMs at 40 is shown in Figure 3.5. For large number of VMs, there is a great dissimilarity in the consumption of energy between the placement strategies by best fit and game theoretic approach. The lower energy consumption is due to the fact shown in Fig. 3.4 that lesser number of PMs are used for dynamic placement by cooperative game theoretic approach than the best fit procedure. Thus, there is a greater save in energy due to the change in the states of PMs from *running* to *idle*.

The results of execution time vs no. of VMs are shown in Fig. 3.6. It is seen that for lesser number of VMs, cooperative game theoretic approach takes lesser time to execute

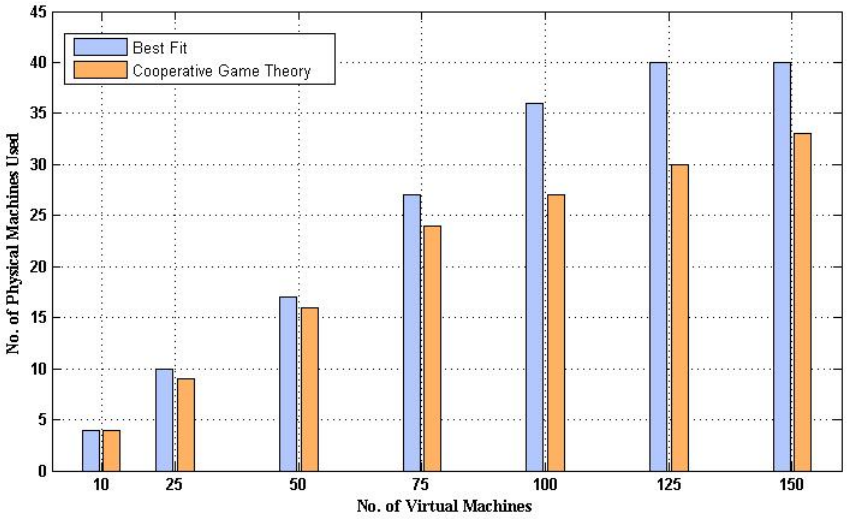


Figure 3.4: No. of PMs Used vs No. of VMs (Available PMs Fixed at 40)

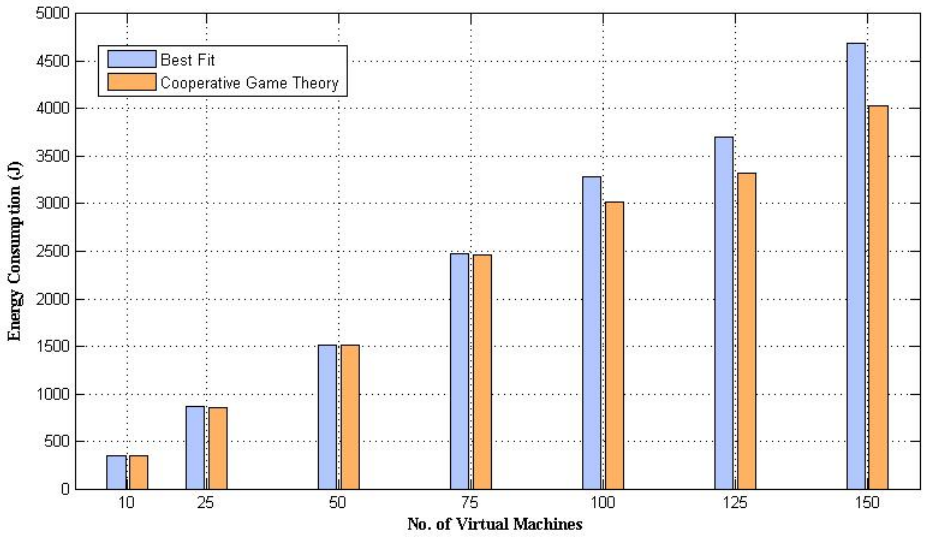


Figure 3.5: Energy Consumption vs No. of VMs (Available PMs Fixed at 40)

than the best fit approach. But as the number of VMs increases, the execution time for the best fit approach becomes lesser. This is due to the fact that the game theoretic approach follows a two-tier optimization strategy; that is first the player chooses the best strategy and then other players selects their strategies optimally and this process continues for all the players which leads to Nash Equilibrium.

The second set of simlations are run considering the fact that number of VMs are fixed at 100 and number of PMs varies from 35-95 in intervals of 15. The lowest number of PMs is taken as 35 since after many tests, it was seen that to place 100 VMs in best fit approach, optimally 35 PMs were at least needed.

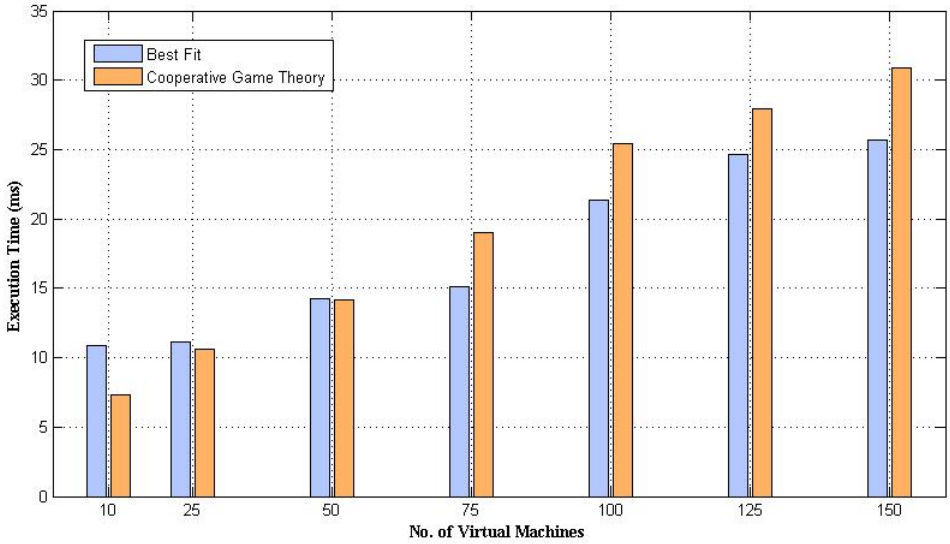


Figure 3.6: Execution Time vs No. of VMs (Available PMs Fixed at 40)

Figure 3.7 shows the experimental results of number of PMs used vs the number of available PMs. It is clearly seen that for all the values, cooperative game theoretic approach yields better results than best fit approach.

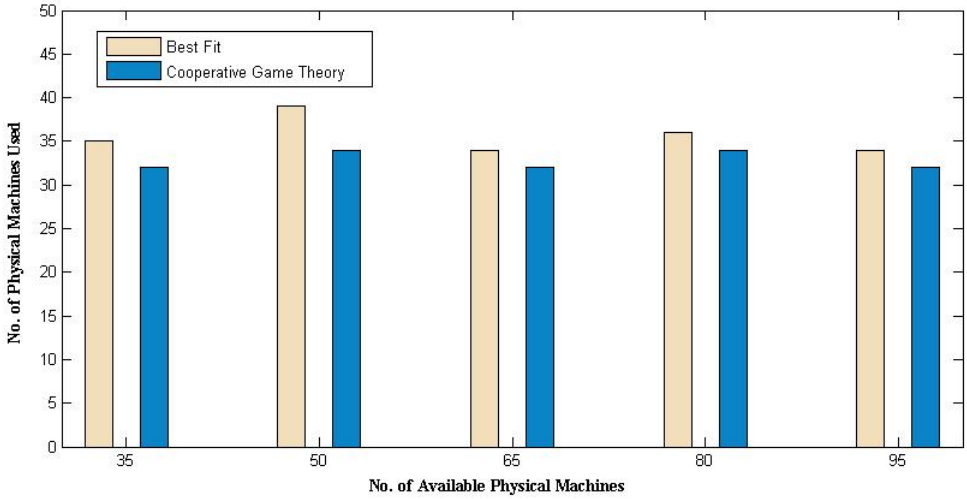


Figure 3.7: No. of PMs used vs No. of Available PMs (VMs Fixed at 100)

The results for energy consumption against number of available PMs are shown in Figure 3.8. Since energy consumption is directly related to the number of PMs used, that is the more the number of PMs in running state, greater is the energy consumption. Since it is seen from fig. 3.7 that game theoretic approach uses lesser number of PMs than the centralized approach to place the VMs, the energy consumption reduces in the decentralized approach.

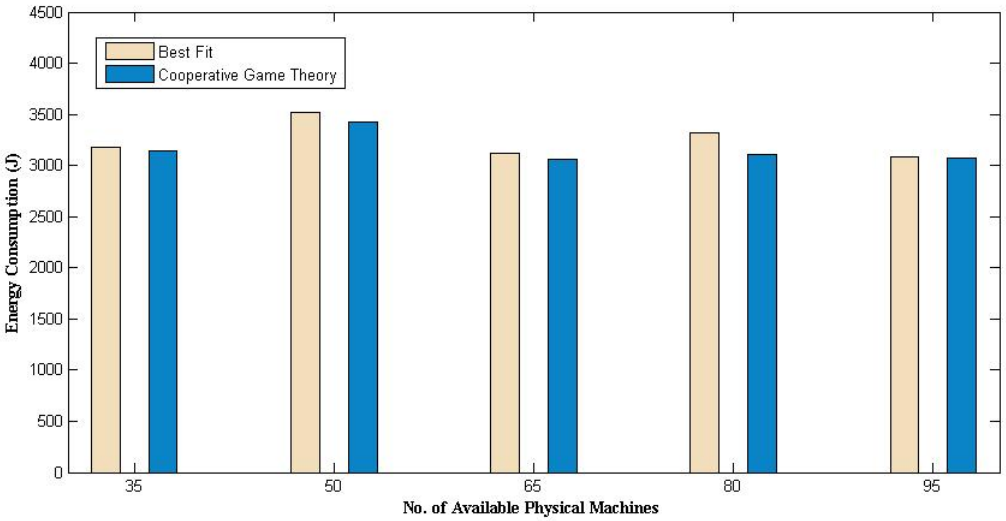


Figure 3.8: Energy Consumption vs No. of Available PMs (VMs Fixed at 100)

The variation of execution time against an increasing number of available PMs is shown in Figure 3.9. Game theoretic approach takes slightly more time for giving the optimal placement strategy for dynamically placing VMs onto PMs, though this fact can be overlooked because the decentralized approach gives a much lesser energy consumption model compared to the best fit approach solution.

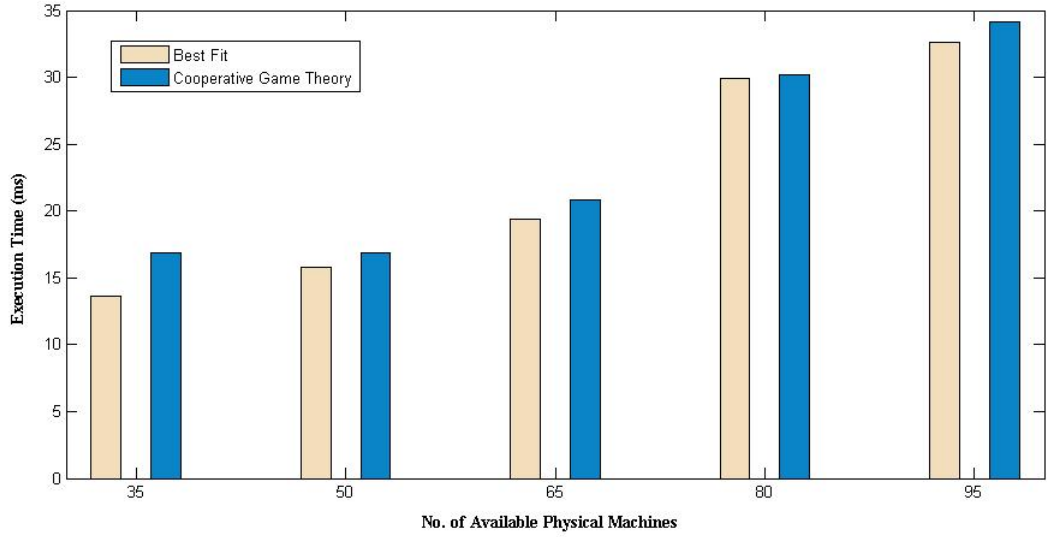


Figure 3.9: Execution Time vs No. of Available PMs (VMs Fixed at 100)

3.9 Conclusion

In this chapter, a novel decentralized procedure to dynamic VM placement problem is given using cooperative congestion game theory. It is shown that the game theoretic approach gives an optimal solution which reaches Nash Equilibrium in polynomial time. The experimental results and the comparisons with centralized approach (best fit algorithm) clearly shows that energy consumption is reduced in game theoretic approach. This overshadows the fact that the execution time is slightly greater in the decentralized model than in the best fit solution.

CHAPTER 4

NON COOPERATIVE GAME THEORY FOR DYNAMIC VM PLACEMENT

4.1 Introduction

In recent times, there has been a growth in the number of cloud users. Thus more number of physical machines are needed in order to process the user requests, which leads to greater energy consumption. As seen in the previous chapter, optimal placement of virtual machines over physical machines is an effective method to minimize the consumption of energy. This goal is not easy to achieve when there is no centralized approach towards decision making and all the decision makers are allowed to act selfishly.

In this chapter, *non-cooperative* approach of game theory is used to arrive at a solution for dynamic virtual machine placement problem. Every decision maker tries to minimize its own energy consumption independently and eventually all of them reach the Nash equilibrium. When the system attains Nash equilibrium, no decision maker can benefit further by altering its own strategy. Thus the main goal of this chapter is to minimize energy consumption in a decentralized cloud computing model by using *non-cooperative game theory*.

4.2 Literature Survey for Non Cooperative Game Theory

Teng et al. in 2010 introduced a novel Bayesian Nash Equilibrium allocation algorithm in order to find solution to the resource management problem in cloud computing (38). It analyzes the bid proportion model where users are assigned resources in proportion to their bids. The allocation solutions reach Nash equilibrium by gambling in stages which

is further simplified by considering that competition among users is for the same job type consisting a sequential order of same type of jobs. In this model, the shortcoming is that bidders fix the resource price and problems of response delay are not not addressed. In the same year, Sun et al. in (36) used continuous double auction and Nash equilibrium for the allocation of resources in cloud based on the M/M/1 queueing system. Sun et al. took performance QOS and economic QOS as optimization objectives. The work in (36) greatly improves the Quality of Service in terms of performance and economy. It is fair to both resources and users by considering execution cost of 'pay per use' services and the average time of execution of user jobs.

The problem of service provisioning was studied by Ardagna et al. in (3). IaaS providers host applications of Saas providers such that every SaaS complies with the SLA (revenue and penalty) requirements. SaaS providers maximize revenues while cost of resources by IaaS is minimized. IaaS providers maximize revenue by providing virtualized resources.The following assumptions for SaaS providers are taken in this work.

- A single web service application is hosted on a single VM.
- Same web service application implemented by many VMs can run in parallel.
- All the virtual machines are uniform in terms of CPU and RAM capacity.

IaaS providers offer flat VMs, on-demand VMs and on-spot VMs for which SaaS providers pay. The service provisioning problem is formulated and solved using game theory. The main drawbacks in this work are listed as follows.

- Validation of the solution is missing by experiments in real world environment.
- The heuristic solutions adopted by IaaS and SaaS providers are not compared.

These drawbacks in 2011 are worked upon by Ardagna et al. and presented in 2013 (4).

Kong et al. (26) worked upon resource allocation among selfish virtual machines by applying stochastic approximation methods. It thwarted non-cooperative behavior of the VMs. The proposed approach is not implemented in a real world virtualized cloud system. A cooperative game theoretic resource splitting solution in cloud computing was formulated by Lu et al. in 2012 (30). Here multiple cloud providers cooperate with one another in order to form a cloud federation which ensures greater profit since provider's capability to serve for public cloud users is enhanced. A game theoretic policy is formed

to aid SPs in the decision making process. The work's performance is not gauged for many cloud environments.

The co-scheduling problem addressed by Dhillon et al. in (10) involves a central authority which provides solution to constrained optimization problem with an objective function. This works provides an alternate game theoretic perspective using stable matchings theory which reduces the Stable Roommates Problem (SRP) to Stable Marriages Problem (SMP). Chen et al. in (9) studied the use of game theory for live migration prediction over cloud computing. It is shown that game theory improves Gilbert-Elliot model for the prediction of the probability on dirty page.

4.3 Observations

Table 4.1: Non Cooperative Game Theory in Distributed Resource Management

Researcher	**Game**	**Environment**	**Work**
2010			
Kołodziej et al. (24)	Non-cooperative	Grid	Combines non-cooperative game theoretic and genetic based meta heuristics to achieve secure task allocation to machines in computational grids.
Teng et al. (38)	Non-cooperative	Cloud	A new Bayesian Nash Equilibrium allocation algorithm is introduced heterogeneous distribution of resources in cloud computing. It analyzes the bid proportion model.
Sun et al. (36)	Non-cooperative	Cloud	Uses Nash equilibrium and continuous double auction for allocation of resources in cloud based on M/M/1 queueing system with the objectives for optimization as economic QOS and performance QOS.

2011			
Ardagna et al. (3)	Non-cooperative	Cloud	Provides an efficient algorithm for the generalized Nash game model to allocate and manage IaaS resources to SaaS components during run time.
Kołodziej et al. (25)	Non-cooperative	Grid	Presents symmetric non-zero game and asymmetric Stackelberg game as non-cooperative game approaches in order to model the requirements of grid users on resource and task allocation.
Kong et al. (26)	Non-cooperative	Cloud	Applies stochastic approximation methods to obtain the stochastic solution of resource allocation problem among selfish virtual machines.
2012			
Buscemi et al. (7)	Non-cooperative	Grid	Propose a non-cooperative grid job scheduling game with the grid sites acting as players and the strategies to be the scheduling algorithms. It focuses on global scheduling.
2013			
Ardagna et al. (4)	Non-cooperative	Cloud	Service provisioning problem is modeled as a generalized Nash game. Two methods based on best-reply dynamics are presented and their solutions approach Nash equilibrium. It is simulated on real prototype environment deployed on Amazon EC2.

Dhillon et al. (10)	Non-cooperative	Cloud	The problem of VM co-scheduling which decreases the performance of virtual machines is provided with a game theoretic perspective by using stable matchings theory.

4.4 Problem Formulation

The system model that is used in this chapter is the same as that used in the previous chapter. The model is shown in Figure 4.1.

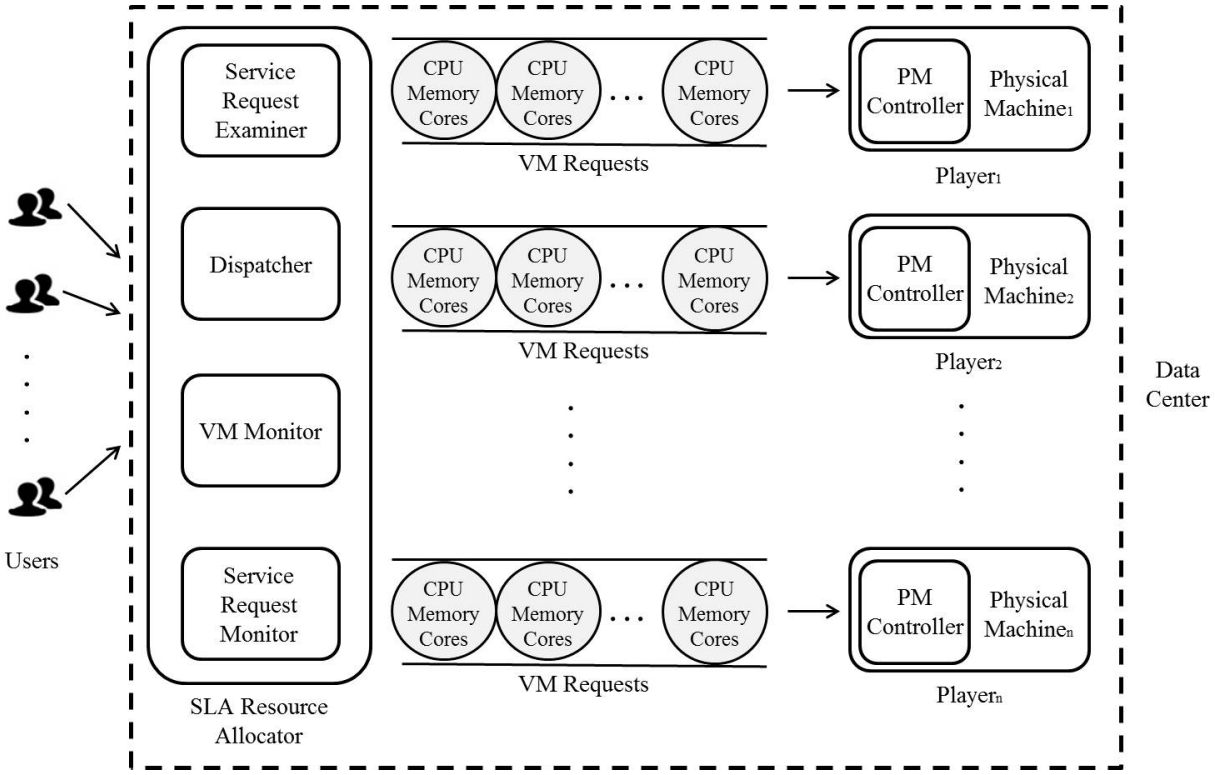


Figure 4.1: Proposed System Model

The major difference between the models in these two chapters is that the physical machines which act as players for the game are non-cooperative here. This means that the physical machines acting as players, will act selfishly and try to minimize its own energy consumption independently. This non-cooperative model has to be solved using non-cooperative game theoretic approach.

In order to arrive at an optimal decision, the decision making process takes help from the *VM Monitor* and the *Service Request Monitor* to get information about the resource

availability and available workload on all physical machines. All physical machines have a PM Controller which act as players for the non-cooperative decision making procedure.

4.4.1 Problem Statement

The dynamic VM placement problem is to be solved with the objective as to minimize energy consumption. In this chapter non-cooperative game theory is used in order to solve the problem since the physical machines acting as players take their own decisions selfishly. Every PM_i has V_i number of VMs which have to optimally placed. In order to form the problem statement, the following arrays are used.

- ***PMS** (1 X P)* - This is used in order to save the present state of the physical machine. Here, 4 states have been considered for every physical machine; 1 = off, 2 = idle, 3 = ready and 4 = running. There are a total of P physical machines.

- ***PMR** (3 X P)* - The array is used to specify the amount of resources every PM has. Here, 3 resources are considered; memory, processor speed and number of cores.

- ***VMR** (3 X V)* - Let V be the total number of VMs in the system; that is combining all PMs. The array is used to indicate the requirements of memory, processor speed and number of cores for each VM.

- ***VMP** (I X V)* - It is considered that in the SLA, I number of performance indicators are mentioned. Then, VMP is used to store the requirement of the performance indicators for all the VMs.

- ***M** (P X V)* - This array is used to store the mapping of PMs to VMs. Thus M_{ij} should be 1 if VM_j is placed in PM_i, else 0.

- ***MP** (I X V)* - In order to store the performance of all the VMs, this array is used. Thus arrays VMP and MP are used to see if the service level agreement is followed or not.

- ***ECS** (P X 4)* - The energy consumption during switching of states is stored here. For PM_i, ECS_{i1} specifies the energy consumption during switching from *off* state to $ready$ state; ECS_{i2}, ECS_{i3}, ECS_{i4} for state switching from $idle$ to $ready$, $ready$ to $idle$ and $idle$ to off respectively. It is considered that there is negligible energy consumption during state transition form $running$ to $ready$.

- ***VMLM*** *(P X P X V)* - This array is used to store the live VM migrations from one PM to another PM.

The energy consumption model is described in chapter 2. The problem statement of dynamic VM placement to minimize energy consumption (VMDPPEC) using non-cooperative game theory can be stated as follows:

$$VMDPPEC = \min TEC(M) \tag{4.4.1}$$

where,

$$TEC(M) = EPS(M) + EVM(M) + EPM(M) \tag{4.4.2}$$

such that,

$$\begin{aligned} &C_{ri} \leq PMR_{ri} \\ &(C_{ri} = VMR \times M'; r = 1, 2, \ldots., R; i = 1, 2, \ldots., P) \end{aligned} \tag{4.4.3}$$

$$\sum_{i=1}^{P} M_{ij} = 1 \qquad (j = 1, 2, \ldots.., V) \tag{4.4.4}$$

$$MP_{iv} \geq VMP_{iv} \;\; (i = 1, 2, \ldots., I; \; v = 1, 2, \ldots.., V) \tag{4.4.5}$$

Equation 4.4.1 shows the optimization objective of the VM dynamic placement problem, i.e. minimization of energy consumption (EC) for a particular placement solution (M) which has to be solved using non-cooperative game theory.

$EPS(M)$ is the total energy consumption by state switching of PMs for a particular solution M.

$EVM(M)$ calculates the total consumption of energy during live VM migrations for the solution M.

$EPM(M)$ gives the result of the total energy consumption of all the PMs in their current states for the solution M.

The solution is arrived upon satisfying all the three constraints as shown in equations 4.4.3, 4.4.4 and 4.4.5. Equation 4.4.3 is the *capacity constraint*, *Placement constraint* is represented in Equation 4.4.4 and Equation 4.4.5 shows that there should be no SLA violations.

4.5 Optimization Algorithm for Non-Cooperative Dynamic VM Placement

In this section, the algorithm based on non-cooperative theory which has been used to solve the problem of dynamic virtual machine placement has been discussed. The symbols used in the algorithm have been listed in Table 4.2.

Table 4.2: List of symbols and their meanings

Symbols	**Meanings**
i	Physical machine number
itr	Iteration number
$energy_i^{(itr)}$	Total energy of i^{th} PM computed at iteration number itr
τ	The accepted tolerance
δ_{p-1}^{α}	A strategy α for PM_{p-1}
en	Tolerance at iteration number itr. $\sum_{i=1}^{n} \left\lvert energy_i^{(itr-1)} - energy_i^{(itr)} \right\rvert$
$command$	The instruction given to the neighbor PM. It has two values, $CEASE$ and $CARRY\_ON$.
$SEND((a, itr, command), i)$	Send message (a, itr, command) to PM_i
$RECEIVE((a, itr, command), i)$	Receive message (a, itr, command) from PM_i
$VMLM\_TEMP$	Temporary live VM migration matrix.
M_TEMP	Temporary VM placement matrix.

Algorithm 4 shows the procedure for a PM to select its best placement every time such that in every iteration the energy consumption by that physical machine is lesser than the energy consumption in the previous iteration for that particular PM. Thus, all the PMs act selfishly in order to optimally dynamically place VMs onto PMs such that energy consumption is minimized.

Algorithm 4: Best_Placement(i)

```
Input: VMLM_TEMP, M_TEMP, itr
Output: energy_i^itr
while (1) do
    Find new placement strategy for PM_i;
    Calculate energy_i^itr
    if (energy_i^itr ≤ energy_i^(itr-1)) then
        Update M_TEMP, VMLM_TEMP;
        return(energy_i^itr);
```

Practically speaking, in order to compute Nash Equilibrium, some coordination must

be present among the players; in this case the physical machines. Thus, the non-cooperative procedure shown in algorithm 5 is devised where players (PMs) are synchronized in such a manner that their individual placement strategies are updated in a round-robin fashion.

Algorithm 5: Non-Cooperative Algorithm

PM i; (i=1, 2, ..., n) executes:

1. Initial :

itr $\leftarrow 0$;
$energy_i^{itr} \leftarrow 0$;
en $\leftarrow 1$;
sum_energy $\leftarrow 0$;
command $\leftarrow CARRY\_ON$;
prev = [(i-2) mod n] + 1;
next = [i mod n] + 1;

2. while (1) **do**

if *(i = 1) // First PM* **then**
prev = n;
if $(itr \neq 0)$ **then**
RECEIVE((en, itr, command), prev);
if $(en \leq \tau)$ **then**
SEND((en, itr, CEASE), next);
exit;
sum_energy $\leftarrow 0$;
itr $\leftarrow itr + 1$;
else
// Other PMs
RECEIVE((sum_energy, itr, command), prev);
if *(command = CEASE)* **then**
if $(i \neq n)$ **then**
SEND((sum_energy, itr, CEASE), next);
exit;
$energy_i^{itr}$ = Best_Placement(VMLM_TEMP, M_TEMP, itr);
sum_energy = sum_energy + $\left|energy_i^{(itr-1)} - energy_i^{(itr)}\right|$;
SEND ((sum_energy, itr, CARRY_ON), next);

Algorithm 5 is executed for all the physical machines periodically or whenever SLA violations are made. When the execution of the algorithm is over, the players will have reached Nash Equilibrium, thus the strategies remain the same in equilibrium.

In this algorithm, the tolerance is maintained at a very small value, such that in every iteration the energy consumption is reduced till a point that very small savings in energy is seen in consecutive iterations. This is the point where the algorithm exits. In order to facilitate the coordination among all the physical machines, messages are sent to the next physical machine. The message consists of three arguments; the energy savings made in that particular iteration, the iteration number and the command. If the command

is $CARRY\_ON$, it means that equilibrium has not been reached and best placement strategy needs to be computed again. On the other hand, the command $CEASE$ instructs the next physical machine that it can exit the algorithm as equilibrium is reached. In every iteration, the first PM checks the tolerance for the last iteration and accordingly initiates the command sequence for all the PMs.

4.6 Simulation and Results

The simulation has been done using an in house simulation using Java. The simulation parameters and the results are given in the next sections.

4.6.1 Simulation Parameters

The series of experiments have been performed considering the capacities of physical machines as each having 12.8 GHz of processor speed, 8 GB memory and is octa-cored. The capacities of VMs are chosen randomly from a fixed set of values.

In order to calculate the consumption of energy by every PM, we use the energy consumption model described in chapter 2. The values for the parameters are shown in Table 4.3. To determine the energy consumption during live VM migration, the parameters have their values fixed; α_i = 0.256, β_i = 10.0825, α_k = 0.256, β_k = 10.0825 and γ_{ik} = 0.5.

Table 4.3: Values for The Energy Consumption Parameters

States	**Running**	**Ready**	**Idle**	**Off**
Running	FE_i(running) = 32.2708	-	-	-
Ready	-	FE_i(ready) = 32.2708	ECS_{i3} = 3.5	-
Idle	-	ECS_{i2} = 7.5	FE_i(idle) = 0.5	ECS_{i4} = 5.5
Off	-	ECS_{i1} = 0	-	FE_i(off) = 0

The Convergence of Nash Equilibrium

In order to prove that the non-cooperative algorithm converges to the nash equilibrium, we have performed experiments; the results of which is shown in Figure 4.2 and Table 4.4. A system of 20 PMs and 50 VMs is considered.

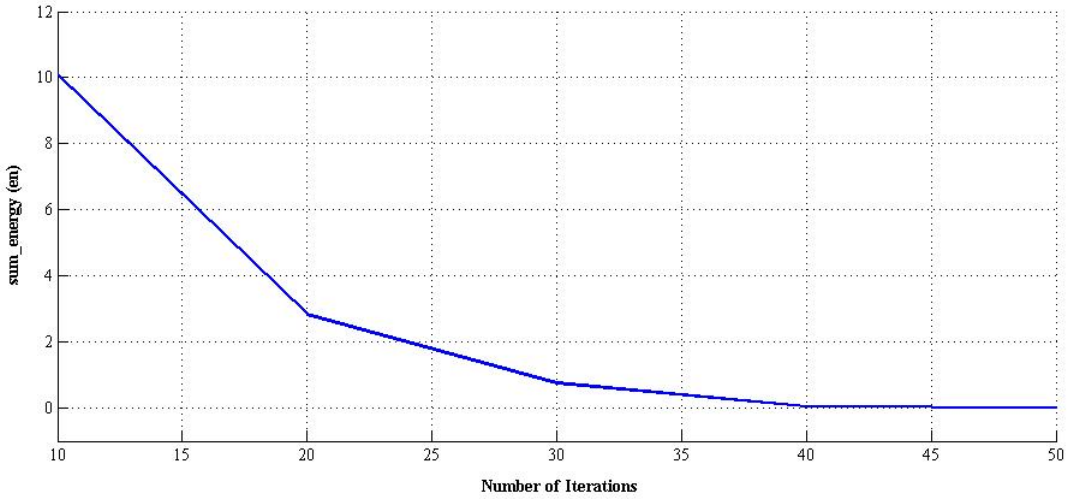


Figure 4.2: en vs No. of Iterations

Table 4.4: sum_energy (en) vs No. of Iterations (for 20 PMs, 50 VMs)

sum_energy (en)	**No. of Iterations**
10.09	10
2.84	20
0.76	30
0.043	40
0.0027	50

It is observed that the value of en converges to zero as the number of iterations increases. This proves that we are achieving Nash Equilibrium in polynomial time. From this experiment, for further simulations, we fix the value of τ as 0.003.

4.6.2 Experimental Results

Figure 4.3 shows the variation of the number of iterations needed to achieve a value for sum_energy less than τ (0.003) as the number of PMs increases. For this simulation, the number of VMs has been fixed at 50.

Next, we perform a set of experiments keeping the number of PMs available fixed at 40 and increasing the number of VMs from 10 to 150. All the results obtained from the non-cooperative game theoretic algorithm are compared with the results obtained from cooperative game theoretic algorithm and best fit algorithm. The impact of selfishness is seen in the results.

The number of PMs used in order to dynamically place the VMs are projected in

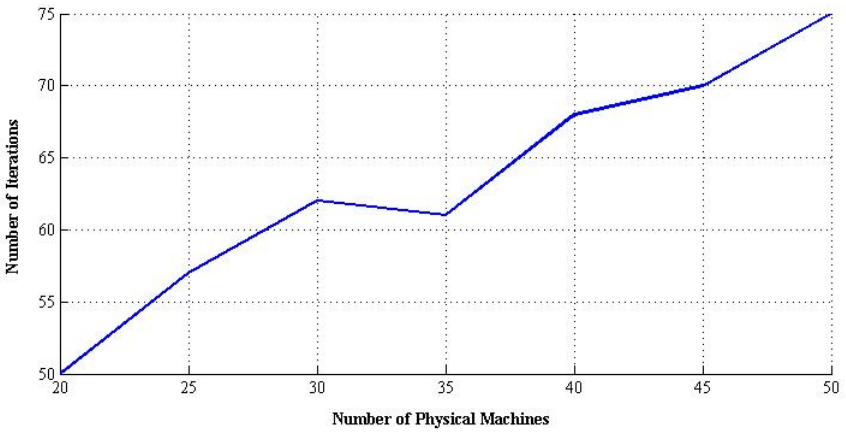


Figure 4.3: Convergence of Non-Cooperative Algorithm (until en $\leq$ 0.003)

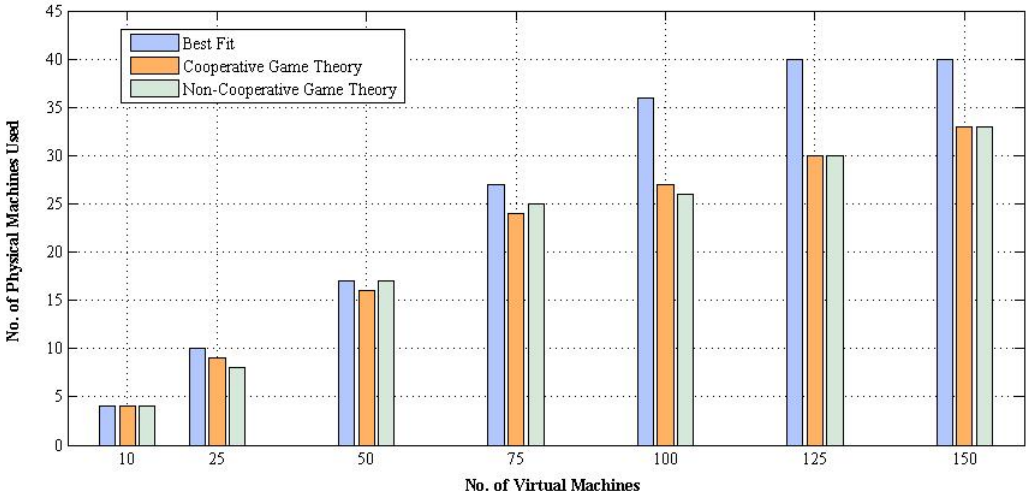


Figure 4.4: No. of PMs Used vs No. of VMs (Available PMs Fixed at 40)

Figure 4.4. The results showing the energy consumption and execution time are shown in Figures 4.5 and 4.6 respectively.

From the results, it is observed that number of PMs used and the energy consumption for the non-cooperative game theoretic algorithm are similar to the results obtained from the cooperative game theoretic approach. Both these algorithms give better results than the best fit approach. But due to the non-cooperative nature of the players, that is PMs, the execution time increases for the non-cooperative approach. But the increase in execution time is not drastic so as to discard the algorithm. For minimization of the energy consumption, a slight increase in the execution time can be afforded.

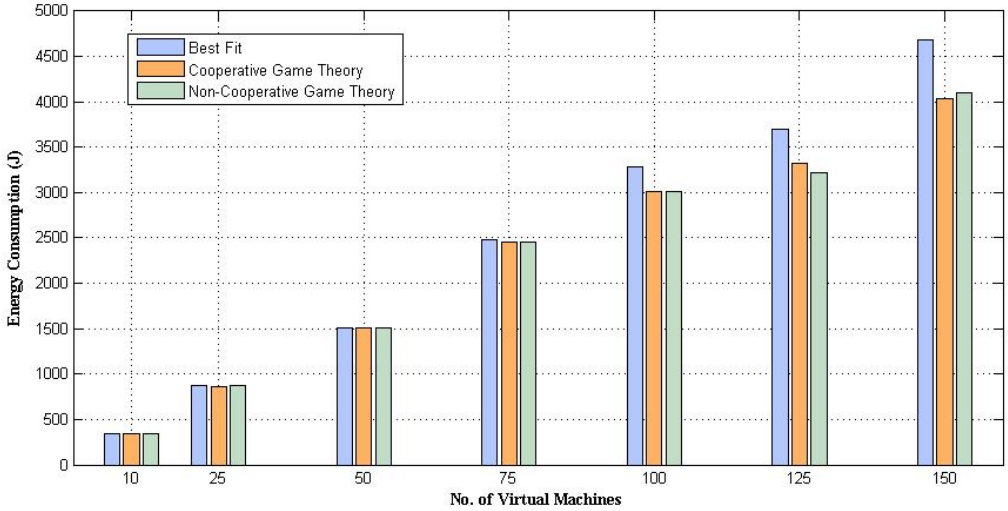


Figure 4.5: Energy Consumption vs No. of VMs (Available PMs Fixed at 40)

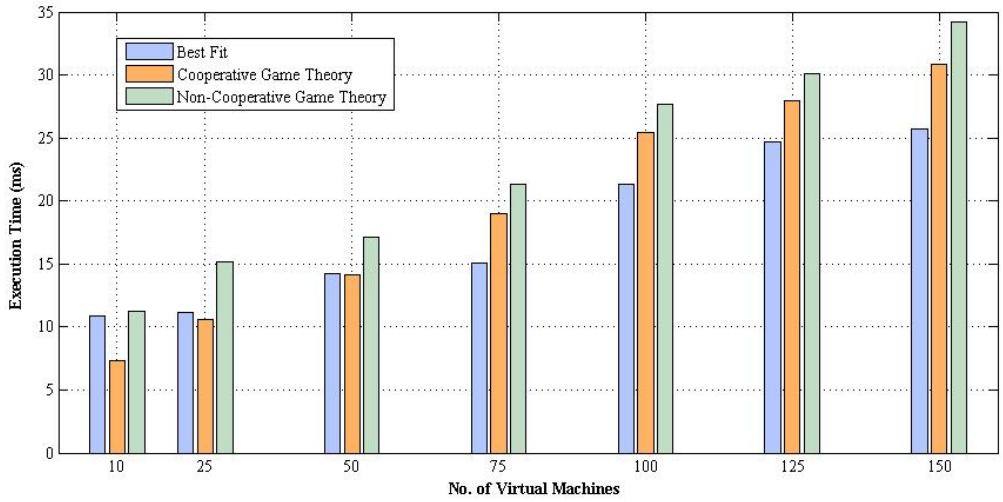


Figure 4.6: Execution Time vs No. of VMs (Available PMs Fixed at 40)

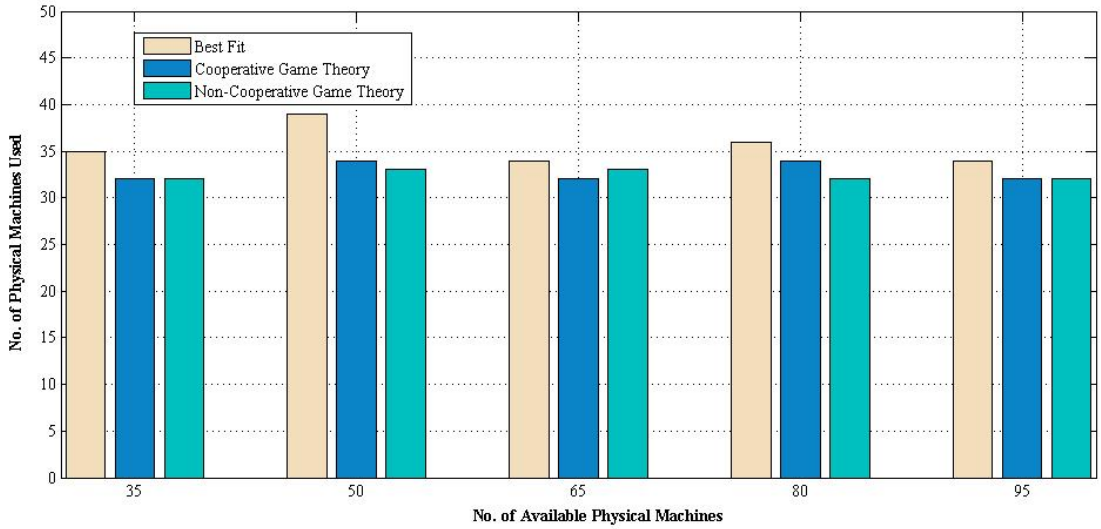


Figure 4.7: No. of PMs used vs No. of Available PMs (VMs Fixed at 100)

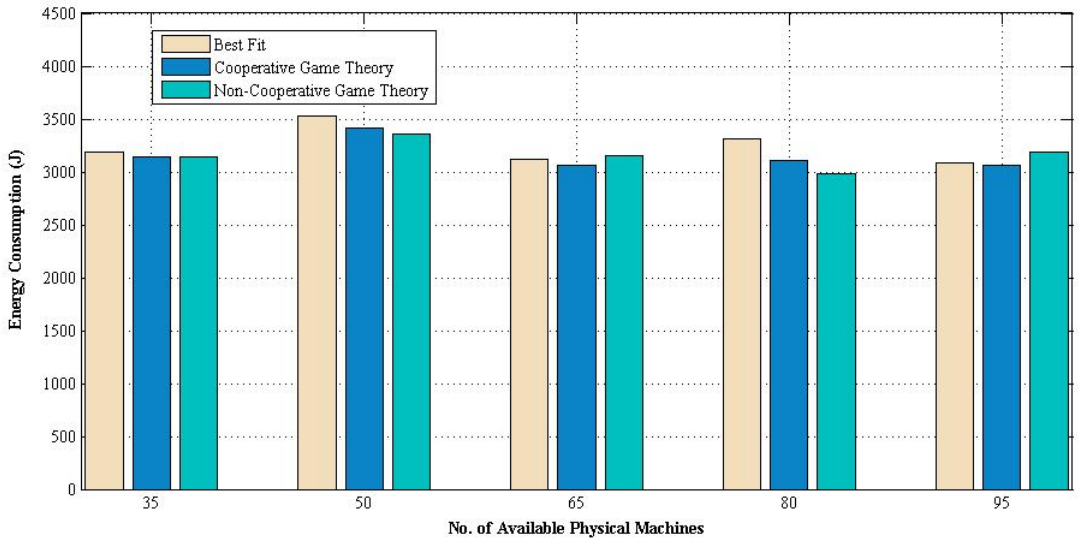


Figure 4.8: Energy Consumption vs No. of Available PMs (VMs Fixed at 100)

The next set of experiments are performed for a number of VMs fixed at 100 while the number of available PMs are increased from 35 to 95 in equal intervals of 15. The results of the non-cooperative approach along with the comparison with the cooperative procedure and the best fit algorithm are shown in Figures 4.7, 4.8 and 4.9.

Here too, the results obtained from the non-cooperative algorithm are similar to the results of cooperative approach which are better than the best fit procedure's results. But the minimization of energy consumption via non-cooperative approach comes at a cost of increased execution time.

Thus the impact of the non-cooperative nature of physical machines affects the execution time to achieve optimal placement of VMs onto PMs while minimizing the energy consumption.

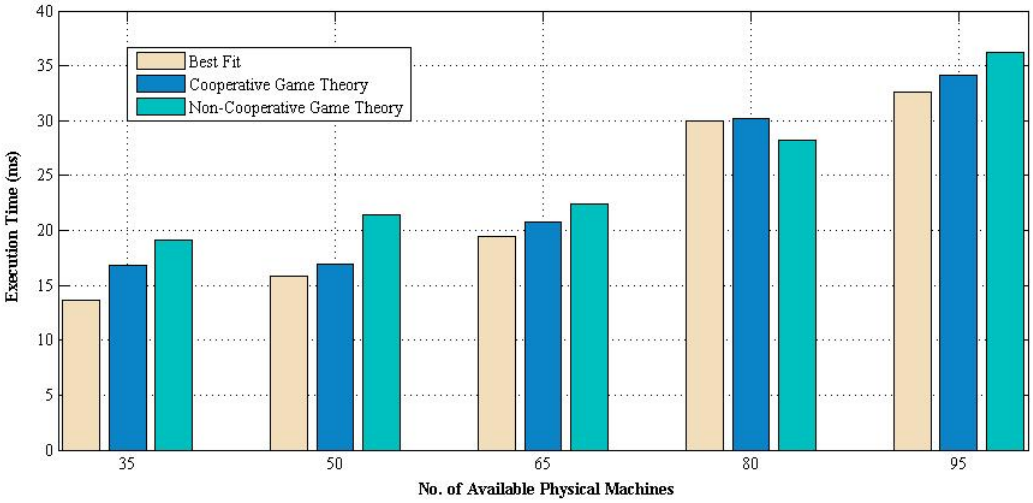


Figure 4.9: Execution Time vs No. of Available PMs (VMs Fixed at 100)

4.7 Conclusion

In this chapter, it is considered that the physical machines do not cooperate with each other in the decision making process for dynamic virtual machine placement. Thus, non-cooperative game theoretic approach is used to find solution to the dynamic virtual machine placement problem. From the experimental results, it can be deduced that the impact of selfishness increases the execution time but the energy consumption minimization is similar to the cooperative game theoretic approach. Also the convergence of Nash equilibrium is shown.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis work, the problem of dynamic virtual machine placement has been addressed. After a deep analysis of the already existing research work in this area, a decentralized approach using game theory has been proposed here in order to optimize the consumption of energy. First, an energy consumption model is built in view of the complicated process of dynamic placement of virtual machines. Here, three factors have been considered while building the model, namely energy consumption of physical machines in different states, consumption of energy during the state transition of physical machines and during live virtual machine migrations. In this work, both cooperative and selfish nature of physical machines have been considered. Cooperative game theory using the concept of congestion game and non-cooperative game theoretic approaches have been used to propose new algorithms to optimally place virtual machines onto physical machines dynamically. For both the approaches, it is seen that Nash equilibrium is achieved in polynomial time. Both the algorithms guarantee a list of live virtual machine migrations to achieve the target solution from the initial mapping. Simulations are done and the experimental results are compared with the best fit approach. Both cooperative and non-cooperative algorithms help in minimizing the energy consumption, though the non-cooperative game theoretic approach takes a longer execution time to give the optimal result.

While conducting the simulations, fixed values are taken for calculating the energy consumption of physical machines. Also it is assumed that all the physical machines are homogeneous in nature. For further research, the values of the parameters for calculation of energy consumption, should be computed dynamically. Also heterogeneous environment should be considered. In addition to these, energy consumption due to other factors like cooling equipment which have not been considered in this work should be taken into account in future.

Bibliography

[1] AL-ALI, R. J., AMIN, K., VON LASZEWSKI, G., RANA, O. F., WALKER, D. W., HATEGAN, M., and ZALUZEC, N., "Analysis and provision of qos for distributed grid applications," *Journal of Grid Computing*, vol. 2, no. 2, pp. 163–182, 2004.

[2] AMOURA, A. K., BAMPIS, E., KENYON, C., and MANOUSSAKIS, Y., "Scheduling independent multiprocessor tasks," in *Algorithms—ESA'97*, pp. 1–12, Springer, 1997.

[3] ARDAGNA, D., PANICUCCI, B., and PASSACANTANDO, M., "A game theoretic formulation of the service provisioning problem in cloud systems," in *Proceedings of the 20th international conference on World wide web*, pp. 177–186, ACM, 2011.

[4] ARDAGNA, D., PANICUCCI, B., and PASSACANTANDO, M., "Generalized nash equilibria for the service provisioning problem in cloud systems," *Services Computing, IEEE Transactions on*, vol. 6, no. 4, pp. 429–442, 2013.

[5] BOBROFF, N., KOCHUT, A., and BEATY, K., "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pp. 119–128, IEEE, 2007.

[6] BORST, S., BOXMA, O., GROOTE, J. F., and MAUW, S., "Task allocation in a multi-server system," *Journal of Scheduling*, vol. 6, no. 5, pp. 423–436, 2003.

[7] BUSCEMI, M. G., MONTANARI, U., and TANEJA, S., "A game-theoretic analysis of grid job scheduling," *Journal of Grid Computing*, vol. 10, no. 3, pp. 501–519, 2012.

[8] CHAISIRI, S., LEE, B.-S., and NIYATO, D., "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pp. 103–110, IEEE, 2009.

[9] CHEN, Y.-L., YANG, Y.-C., and LEE, W.-T., "The study of using game theory for live migration prediction over cloud computing," in *Intelligent Data analysis and its Applications, Volume II*, pp. 417–425, Springer, 2014.

[10] DHILLON, J. S., PURINI, S., and KASHYAP, S., "Virtual machine coscheduling: A game theoretic approach," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pp. 227–234, IEEE, 2013.

[11] DO, A. V., CHEN, J., WANG, C., LEE, Y. C., ZOMAYA, A. Y., and ZHOU, B. B., "Profiling applications for virtual machine placement in clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 660–667, IEEE, 2011.

[12] DUPONT, C., GIULIANI, G., HERMENIER, F., SCHULZE, T., and SOMOV, A., "An energy aware framework for virtual machine placement in cloud federated data centres," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pp. 1–10, IEEE, 2012.

[13] GE, Y., ZHANG, Y., QIU, Q., and LU, Y.-H., "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pp. 279–284, ACM, 2012.

[14] HASSAN, M. M., SONG, B., and HUH, E.-N., "Game-based distributed resource allocation in horizontal dynamic cloud federation platform," in *Algorithms and Architectures for Parallel Processing*, pp. 194–205, Springer, 2011.

[15] HERMENIER, F., LORCA, X., MENAUD, J.-M., MULLER, G., and LAWALL, J., "Entropy: a consolidation manager for clusters," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 41–50, ACM, 2009.

[16] HERMENIER, F., LORCA, X., MENAUD, J.-M., MULLER, G., and LAWALL, J., "Entropy: a consolidation manager for clusters," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 41–50, ACM, 2009.

[17] HOTZ, H., "A short introduction to game theory," 2006.

[18] HYSER, C., MCKEE, B., GARDNER, R., and WATSON, B. J., "Autonomic virtual machine placement in the data center," 2008.

[19] JACKSON, M. O., "A brief introduction to the basics of game theory," 2011.

[20] JONES, M. T., "Anatomy of an open source cloud," *IBM developerWorks. Available: http://www. ibm. com/developerworks/opensource/library/oscloud-anatomy. Visited on May*, 2010.

[21] KANTARCI, B., FOSCHINI, L., CORRADI, A., and MOUFTAH, H. T., "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 708–713, IEEE, 2012.

[22] KHAZAEI, H., MISIC, J., and MISIC, V. B., "Modelling of cloud computing centers using m/g/m queues," in *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pp. 87–92, IEEE, 2011.

[23] KHAZAEI, H., MISIC, J., and MISIC, V. B., "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 936–943, 2012.

[24] KOŁODZIEJ, J. and XHAFA, F., "A game-theoretic and hybrid genetic meta-heuristic model for security-assured scheduling of independent jobs in computational grids," *Proc. of CISIS*, pp. 93–100, 2010.

[25] KOŁODZIEJ, J. and XHAFA, F., "Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids," *International Journal of Applied Mathematics and Computer Science*, vol. 21, no. 2, pp. 243–257, 2011.

[26] KONG, Z., XU, C.-Z., and GUO, M., "Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 614–621, IEEE, 2011.

[27] LEE, C., SUZUKI, J., VASILAKOS, A., YAMAMOTO, Y., and OBA, K., "An evolutionary game theoretic approach to adaptive and stable application deployment in clouds," in *Proceedings of the 2nd workshop on Bio-inspired algorithms for distributed systems*, pp. 29–38, ACM, 2010.

[28] LIAO, X., JIN, H., and LIU, H., "Towards a green cluster through dynamic remapping of virtual machines," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 469–477, 2012.

[29] LONDOÑO, J., BESTAVROS, A., and TENG, S.-H., "Collocation games and their application to distributed resource management," tech. rep., Boston University Computer Science Department, 2009.

[30] LU, Z., WEN, X., and SUN, Y., "A game theory based resource sharing scheme in cloud computing environment," in *Information and Communication Technologies (WICT), 2012 World Congress on*, pp. 1097–1102, IEEE, 2012.

[31] MAO, Z., YANG, J., SHANG, Y., LIU, C., and CHEN, J., "A game theory of cloud service deployment," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, pp. 436–443, IEEE, 2013.

[32] MILCHTAICH, I., "Congestion games with player-specific payoff functions," *Games and economic behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[33] NIYATO, D., VASILAKOS, A. V., and KUN, Z., "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 215–224, IEEE Computer Society, 2011.

[34] NIYATO, D., ZHU, K., and WANG, P., "Cooperative virtual machine management for multi-organization cloud computing environment," in *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 528–537, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[35] PILLAI, P. S. and RAO, S., "Resource allocation in cloud computing using the uncertainty principle of game theory,"

[36] SUN, D., CHANG, G., WANG, C., XIONG, Y., and WANG, X., "Efficient nash equilibrium based cloud resource allocation by using a continuous double auction," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, vol. 1, pp. V1–94, IEEE, 2010.

[37] SUN, W., ZHANG, D., ZHANG, N., ZHANG, Q., and QIU, T., "Group participation game strategy for resource allocation in cloud computing," in *Network and Parallel Computing*, pp. 294–305, Springer, 2014.

[38] TENG, F. and MAGOULÈS, F., "A new game theoretical resource allocation algorithm for cloud computing," in *Advances in Grid and Pervasive Computing*, pp. 321–330, Springer, 2010.

[39] THIRUVENKADAM, M. T. and KARTHIKEYANI, V., "An approach to virtual machine placement problem in a datacenter environment based on overloaded resource," 2014.

[40] TUROCY, T. L. and VON STENGEL, B., "Game theory*: Draft prepared for the enciclopedia of information systems," tech. rep., CDAM Research Report LSE-CDAM-2001-09, 2001.

[41] URGAONKAR, B., ROSENBERG, A. L., and SHENOY, P., "Application placement on a cluster of servers," *International Journal of Foundations of Computer Science*, vol. 18, no. 05, pp. 1023–1041, 2007.

[42] WEI, G., VASILAKOS, A. V., ZHENG, Y., and XIONG, N., "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[43] WOOD, T., SHENOY, P. J., VENKATARAMANI, A., and YOUSIF, M. S., "Black-box and gray-box strategies for virtual machine migration.," in *NSDI*, vol. 7, pp. 17–17, 2007.

[44] XHAFA, F. and KOLODZIEJ, J., "Game-theoretic, market and meta-heuristics approaches for modelling scheduling and resource allocation in grid systems," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pp. 235–242, IEEE, 2010.

[45] XIAO, Z., JIANG, J., ZHU, Y., MING, Z., ZHONG, S., and CAI, S., "A solution of dynamic vms placement problem for energy consumption optimization based on evolutionary game theory," *Journal of Systems and Software*, vol. 101, pp. 260–272, 2015.

[46] XU, J. and FORTES, J. A., "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 179–188, IEEE, 2010.

Dissemination

Published

1. Arnab Kumar Paul, Sourav Kanti Addya, Bibhudatta Sahoo, and Ashok Kumar Turuk, ”Application of Greedy Algorithms to Virtual Machine Distribution across Data Centers” in 2014 Annual IEEE India Conference (INDICON), pp. 1-6, IEEE, 2014.