On-Demand VM Placement on Cloud Infrastructure

Sameer Kumar Mandal

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

On-Demand VM Placement on Cloud Infrastructure

Dissertation submitted for the partial fulfillment of

the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

Specialization: Computer Science

by

Sameer Kumar Mandal

(Roll 211CS1071)

under the supervision of

Prof. Pabitra Mohan Khilar

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India

Dedicated to Shri Sai Baba & to my Parents

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

June 05, 2013

Certificate

This is to certify that the work in the thesis entitled *On-Demand VM Placement on Cloud Infrastructure* by *Sameer Kumar Mandal*, bearing roll number *211CS1071*, 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*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Pabitra Mohan Khilar

Acknowledgment

First and foremost I would like to thank Almighty GOD, the compassionate, the almighty merciful, who kindly helped me to complete this thesis.

I would like to express my deep sense of respect and gratitude towards my supervisor Prof. Pabitra Mohan Khilar, who has been the guiding force behind this work. I want to thank him for introducing me to the field of Cloud Computing and giving me the opportunity to work under him. Without his invaluable advice and assistance it would not have been possible for me to complete this thesis. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I would like to thank all the professors, co-researchers, batch mates and friends at National Institute of Technology Rourkela for their active or hidden cooperation.

I thank all the members of the Department of Computer Science and Engineering, and the Institute, who helped me by providing the necessary resources, and in various other ways, in the completion of my work.

Finally, I want to dedicate this thesis to my parents, my family members and my beloved one for their unlimited support and strength. Without their dedication and dependability, I could not have pursued my M.Tech degree at the National Institute of Technology Rourkela.

Sameer Kumar Mandal

Abstract

Cloud Computing paradigm is most popular because of its flexibility for provisioning resources quickly and efficiently. In cloud computing the resource requests are served by creating virtual machines of the requested specification on the underlying physical infrastructure. If the placement of virtual machines to the underlying physical machines will take long time or if all the accepted virtual machine requests can't be served then some flexibility will lost. In on-demand access to cloud computing services the requested resources are served on the available infrastructure for short span of time. In on-demand access the number of resource requests in a particular time interval can not be predicted unlike in case of spot-market access. As a virtual machine instance will run on a single physical machine at a time, hence to serve more requests in case of on-demand access we have to use the available resource optimally considering the allocation cost and SLA violation. In this work we tried to improve the resource utilization by considering single dimensional best fit strategy, which not only reduce the cost by utilizing minimum number of resources but also minimize the SLA violation which may arise due to failure in allocating all the requested virtual machine. We have developed a framework that optimizes the use of physical infrastructure by effectively allocating the requested virtual machines and also reduces the allocation time. The proposed allocation policy is compared with three other existing policies named Greedy First Fit, Ranking and Round-Robin, by simulating all policies using CloudSim toolkit and the performance is evaluated by considering various parameters.

***Keywords*:** Cloud Computing, Virtual Machine Placement, Resource Allocation, Resource Utilization, Virtual Machine scheduling, Minimizing Resource Usages, Minimizing SLA Violation, On-demand Access.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduction

Resource Allocation in Cloud Computing

Motivation

Defining the Problem Statement

Structure of the Thesis

Chapter 1

Introduction

1.1 Introduction

Cloud computing is the utility computing that has unlimited virtualized resource to create a custom-built infrastructure or platform to run applications or full part services as pay-as-you-use basis. Cloud computing has modified the paradigm of system deployment [2]. The advanced system implementation are abstracted from the end user by the help of the virtualization techniques, resources are virtualized that offers an illusion of infinitely scalable and universally available system [1] [2]. With the assistance of pay-as-you-use model, infinitely scalable and universally available systems, cloud computing makes the long command dream of utility computing possible. Developers having initiative concepts for providing new Internet services do not need massive expenditure to setup the hardware and software necessities. Cloud computing refers to each application that are provided as a service over the Internet and therefore the underlying hardware infrastructure and the platform over that those applications are developed. The underlying hardware infrastructure is technically called data-center, having massive vary of physical devices starting from personal computers to cluster computers to high end server machines. [3].

It is an imprudently taken conception by many of us that cloud computing is nothing however the Internet given a special name. Typically this can be as a result of several drawing of web based application often drawn as a cloud. Cloud computing is an abstraction supported the assumption of pooling physical resource and representing them as virtual resource. This new model offers provisioning of resources for staging application and for platform freelance user access to services. Once you contemplate the event of cloud computing up to now, it's clear that the technology is the results of the convergence of many totally different standards. Cloud computings promise of measurability utterly changes the way during which services and applications are deployed. While not standards, the trade creates proprietary systems with merchandiser lock-in. As a result shoppers don't need to be fast into any single system, there's a robust trade push to make standards-based clouds. The cloud computing trade is functioning with these subject standards: Platform virtualization of resources, Service-oriented design, Web-application frameworks, Preparation of open-source software system, Standardized Internet services, Involuntary systems. Cloud computing is especially valuable as a result of it shifts capital expenditures into operation expenditures. This has the good thing about decoupling growth from money handy or from requiring access to capital. It additionally shifts risk from a company and onto the cloud service provider. Service Level Agreements (SLAs) is a vital side of cloud computing, they're primarily your operating contract with any supplier [2].

A major a part of cloud computings price proposition and its charm is its ability to convert capital expenses to operational expenses through a usage rating theme that's elastic and might be right sized. The conversion of real assets to virtual ones provides a live of protection against an excessive amount of or deficient infrastructure. Basically, moving expenses onto the operational expenses facet of budget permits a company to transfer risk to cloud computing supplier.

The chapter 1 is organized as follows. In section 1.2 resource allocation in cloud computing is discussed. In section 1.3 the motivation for this work is discussed. In

section 1.4 the Problem statement is discussed. In section 1.5 the outline of this thesis is presented.

1.2 Resource Allocation in Cloud Computing

Resource allocation is a topic that has been addressed in several computing areas, like operating systems, grid computing, and datacenter management. A Resource Allocation System in Cloud Computing are often seen as any mechanism that aims to ensure that the application's needs are attended properly by the provider's infrastructure. In conjunction with this guarantee to the developer, resource allocation mechanisms ought to additionally take into account the present status of every resource within the cloud environment, so as to use algorithms to better allocate physical and/or virtual resources to developers' applications, therefore minimizing the operational cost of the cloud environment .

Allocation of virtual machines are often divided in two parts: The first part is admission of latest requests for virtual machines provisioning and placement virtual machines on hosts, whereas the second part is optimization of current allocation of virtual machines. The complexity of the allocation a part of the algorithm is $n \times m$, where n is the number of virtual machines that have to be allotted and m is the number of hosts. Optimization of current allocation of VMs is disbursed in 2 steps: at the first step virtual machines that require to be migrated are choosen, at the second step chosen virtual machines are placed on the host machine by allocation algorithm [4].

The virtual machine placement algorithm ensures the efficient and cost effective allocation of virtual machines on the actual host machines.

1.3 Motivation

Cloud Computing paradigm is most popular because of its flexibility for provisioning resources quickly and efficiently. In Infrastructure as a Service (IaaS) cloud computing the resource requests are served by creating virtual machines of the requested specification on the underlying physical infrastructure. If the placement of virtual machines to the underlying physical machines will take long time or if all the accepted virtual machine requests can't be served then some flexibility will lost. In case of on-demand access to cloud computing services the requested resource are served on the available infrastructure for short span of time. In on-demand access the number of resource requests in a particular time interval can not be predicted unlike in case of spot-market access. As a virtual machine instance will run on a single host at a time, hence to serve more requests in case of on-demand access we have to use the available resource optimally.

Most of the work have been done focusing on reduction of allocation cost only. But along with the allocation cost we have to consider whether all the requested virtual machines are getting allocated and whether the service provider is able to serve more request in that particular time-frame (i.e. SLA violation should be taken into consideration). This can be done by allocating a virtual machine to a physical host that best fit the requirements, instead of simply allocating to a host that can serve the request. So an efficient virtual machine placement framework should be developed that can not only allocate the virtual machines using less no of host machines (cost-effective allocation) but also can serve maximum number of requests (minimizing SLA violations).

1.4 The Problem Statement

Our problem statement can be briefly described as follows:

- Let $Host_{List} = \{H_1, H_2, H_3, H_n\}$ is the list of physical hosts available which

have to process the virtual machine requests.

- The virtual machine requests that have been made in a particular time interval are collected at the virtual machine manager and are stored in a VM_{Queue}, $VM_{Queue} = \{VM_1, VM_2, VM_3, VM_n\}$.
- Here we have to do a mapping $VM_{Queue} \rightarrow Host_{List}$, the mapping should be done in such a way that it will minimize the use of resources so that more number of requests can be served with the available resource capacity.
- The aim is to reduce the cost of allocation and minimize the SLA violation.

1.5 Structure of The Thesis

The rest of the thesis is organized as follows. Chapter 2 gives an overview of cloud computing, which includes the terminologies and technologies used in cloud computing. Chapter 3 describes the works that have been done addressing the issue of virtual machine placement along with the strategies followed by open-source technologies. In Chapter 4 the proposed framework for virtual machine placemet as well as the VM Scheduler algorithm is described. Chapter 5 presents the simulation and results of the proposed framework. Chapter 6 concludes the discussion and gives a direction to future research in this issue we have discussed.

1.6 Conclusion

Resource Allocation System in Cloud Computing are often seen as any mechanism that aims to ensure that the application's needs are attended properly by the provider's infrastructure. In case of on-demand access to cloud computing services the requested resource are served on the available infrastructure for short span of time.In this thesis an efficient virtual machine placement framework has been developed that can not only allocate the virtual machines using less no of host

machines (cost-effective allocation) but also can serve maximum number of requests (minimizing SLA violations).

Chapter 2

An Overview of Cloud Computing

Defining Cloud Computing

Deployment Models

Service models

Virtualization vs. Cloud Computing

Virtual Machine and Cloud Computing

Hypervisor in Cloud Computing

Open-source Cloud Computing Solutions

Chapter 2

An Overview of Cloud Computing

2.1 Introduction

There is plenty of dialogue of what cloud computing is. The US National Institute of Standards and Technologies (NIST) has place a shot in process cloud computing. According to NIST [5] *Cloud computing is a model for convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

The above definition is often explained briefly as, network access in on-demand basic and in a very convenient manner along with less effort from management and less service provider's interaction explains quick and straight forward access for potential resources. With resources in a shared pool, illustrates the supply of computing resources from a cloud service provider are combined in a one massive assortment, for serving all users. The frequent provisioning of resources is employed for quickly matching the active resources, once a necessities comes for those resources. This frequent and quick provisioning prevents a scarcity of computing power once the requirement will increase. .

Cloud computing takes the technology, services, and applications that are almost

like those on the Internet and turns them into a self-service utility. The use of the word cloud makes relevancy to two essential ideas:

1. Abstraction: Cloud computing abstracts the complexcity of system implementation from developers and users. Applications run on physical systems that aren't nominative, data is hold on in locations that are unknown, administration of system is outsourced to others, and access by users is present.

2. Virtualization: Cloud computing virtualizes system by pooling and sharing resources. Systems and storage may be provisioned as required from a centralized infrastructure, multi-tenancy is enabled, prices are assessed on a metered basis, and resources are ascendible with lissomeness.

What does cloud agility mean? Its tied to the fast provisioning of computing resources. Cloud environments usually give new compute instances or storage in minute [6]. As one may imagine, the dramatic shortening of the provisioning time-frame permits work to begin way more quickly.

What does multi-tenancy means? It refers to principal of software design where one instance of the software runs on a server, serving multiple shopper organizations (tenants). During a multi-tenancy setting, multiple customers share a similar application, running similar software, on a similar hardware, with a similar data-storage mechanism [7].

To discuss cloud computing in detail, we would like to outline the lexicon of cloud computing; The majority separate cloud computing into 2 distinct sets of models:

Deployment models: This refers to the placement and management of the cloud infrastructure.

Service models: This consists of the varieties of services that you can access on a cloud computing platform.

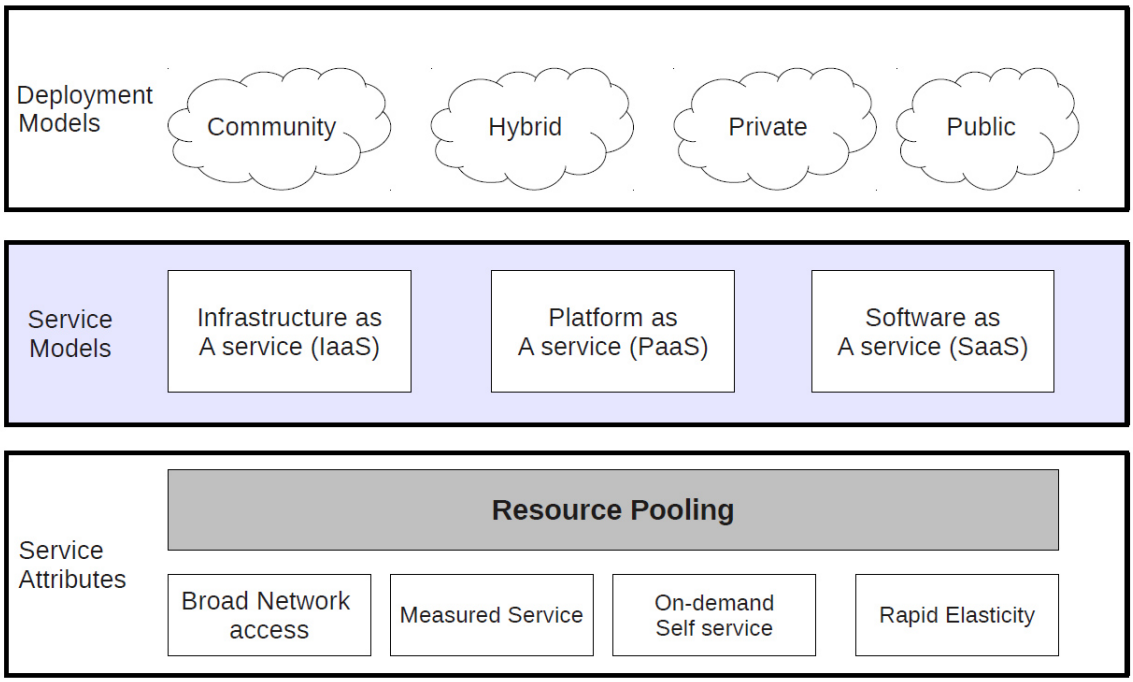

Figure 2.1: The NIST model of cloud computing

The chapter 2 is organized as follows. In section 2.2 the deployment models are discussed. In section 2.3 service models are discussed. In section 2.4 virtualization vs. cloud computing. In section 2.5 virtual machine and cloud computing is discussed. In section 2.6 hypervisor in cloud computing is discussed. In section 2.7 Open-source Cloud Computing Solutions are discussed.

2.2 Deployment Models

Deployment model defines the aim and location of the cloud. Based on the ownership the cloud is classified into four deployment model [5] private cloud, public cloud, community cloud and hybrid cloud.

2.2.1 Private Cloud

Private cloud (also referred to as internal cloud or company cloud) is a term for a proprietary computing design that gives hosted services to a restricted variety of individuals behind a firewall. The cloud could be managed by that organization or a third party. Private cloud could also be either on or off premises. This offers organization the price advantages of virtualization.

2.2.2 Public Cloud

A public cloud is one that supports the quality cloud computing model, during which a service provider makes resources, like applications and storage, out there to the public over the web. Public cloud services are free or offered on a pay-per-usage model. It's an extension of private cloud with further value profit owing to service provider. The most edges of employing a public cloud service are; simple and cheap set-up, quantifiable to fulfill wants, no wasted resources as a result of to acquire what you utilize.

2.2.3 Hybrid Cloud

A hybrid cloud is a composition of a minimum of one private cloud and a minimum of one public cloud. A hybrid cloud is usually offered in one amongst two ways: a seller incorporates a private cloud and forms a partnership with a public cloud supplier, or a public cloud supplier forms a partnership with a seller that gives private cloud platforms. A hybrid cloud is a cloud computing model within which a corporation provides and manages some resources in-house and has others provided outwardly. Ideally, the hybrid approach permits a business to require advantage of the measurability and cost-effectiveness that a public cloud computing setting offers while not exposing mission important applications and data to third-party.

2.2.4 Community Cloud

A community cloud is a multi-tenant infrastructure that's shred among many organizations from a selected cluster with common computing issues. Such issues may be well associated with regulative compliance like audit necessities, or is relate to performance necessities, like hosting applications that need a fast latent period. The community cloud are often either on-premises or off-premises, and may be ruled by the taking part organization or by third party managed service provider.

2.3 Service models

After deployment of the cloud, completely different vendors provide clouds that have different services related to them. The portfolio of service offered adds another set of definitions referred to the service model. There are various services models delineated within the literature, all of that take the shape *XaaS* or *<Something >as a Service* [2].
Three service types [5] have been universally accepted:

2.3.1 Software as a Service (SaaS)

Software as a Service provides a whole environment with pre-installed applications along side infrastructue. Consumer will access these applications by using any device capable of operate an Internet browser. This can be supported the thought of dealings software from a service supplier than shopping for itself. The software is hosted on centralized network servers to create practicality offered over the Internet or any other network. Additionally called Software on demand it's presently most popular style of cloud computing, attributable to its high flexibility, nice services, increased measurability and fewer maintenance e.g., Yahoo mail, Google docs, CRM client knowledge. SaaS is extremely effective in lowering the prices of business because it provides access to applications at a price ordinarily way cheaper than

a commissioned application fee, that is feasible because of its monthly fees based revenue model. With SaaS user needn't worry concerning installation or upgrades.

2.3.2 Platform as a Service (PaaS)

Platform as a Service provides a framework for the developers to create and deploy their own application on a hosted infrastructure. The client doesn't have to be trouble concerning the underlying hardware; they solely have to be compelled to manage the in-operation environment with the interface provided to them. They can use any artificial language supported by the cloud service provider to create their application in that environment e.g., SalesForce.coms Force.com [8].

2.3.3 Infrastructure as a Service (IaaS)

In Infrastructure as a Service computing model consumer will borrow the basic hardware resources for building their own framework. They'll customize their entire framework with the assistance of virtual machines, memory, virtual network etc. The consumer doesn't have to manage the physical resources, as they're supplied with virtual resources which can be managed programmatically. It delivers the computing infrastructure as a totally outsourced service. A number of the businesses that give infrastructure as a service are Google, IBM, Amazon.com etc.Virtualization allows IaaS suppliers to supply virtually unlimited instances of servers to customers and build efficient use of the hosting hardware.

2.4 Virtualization vs. Cloud Computing

Virtualization and cloud computing were each developed to maximise the employment of computing resources whereas streamlining processes and increasing efficiencies to cut back the full value of possession.

Virtualization could be accustomed to offer cloud computing, cloud computing

is sort of completely different from virtualization. Cloud computing to most people might appear as if virtualization and is incredibly similar in fashion, however it is often higher represented as a service where virtualization is a component of a physical infrastructure [9].

Cloud computing was born from the construct of utility computing. Utility computing was the idea that computing resources and hardware would become a trade goods to the corporations and/or federal agencies would purchase computing resources from a central pool and pay just for resource they used. These resources would be metered, such as you get power for your home from an electric company.

The real distinction between virtualization and cloud computing is truly not that troublesome to grasp. For example, a self service model isn't a vital element in virtualization, however is in cloud computing. You'll actually argue some virtualization solutions might embrace a self service component; but, it's not obligatory. In cloud computing, self-service could be a crucial construct to deliver accessibility to any user at any time, that is what a service is all regarding. Moreover, self-service is an efficient mechanism to cut back the number of coaching and support required in the slightest degree levels among a company. Both technologies will prevent cost. If you select to use virtualization, you may incur a good deal of direct value however will save additional on operational expenditures within the long-standing time. Cloud computing works simply in an opposite fashion, you would not like several resources at the start, therefore cloud computing can seemingly value little within the starting. However, as your applications become additional fashionable to your users, the demand on resources will increase.

2.5 Virtual Machine and Cloud Computing

Virtual machines includes virtual hardware and a real software package.The virtual machines give an entire setting for application to run similar to they might on their own individual server, together with each the hardware and software package [10].

Enterprise needs are driving the evolution and adoption of the cloud and this can create the utilization of virtual machines even additional vital than it's been up to now.

In Infrastructure as a Service model the infrastructure requests are served by allocating virtual machines to those requests [2] [11]. Therefore on service suppliers aspect one in every of the first concern is the allocation of virtual machine on the particular physical resource i.e. the underlying hardware infrastructure. This allocation ought to be economical in order that resource utilization are optimized and to serve in lesser time.

The virtual machine placement in cloud computing are often created by considering any one of the three classes [12]; reservation, on-demand access and spot markets. In case of reservation user must pay a particular fee for a selected amount for every instance of a virtual machine. In case of on-demand access user requests virtual machine for immediate access for a relatively short interval of time, and pay the charge relying upon that period. In case of spot markets the users ought to specify the price they're willing to buy the requested virtual machines, as in spot markets there's frequent fluctuation in providers value. The users are allotted with virtual machines only if the providers value is same or less than the users fee.

2.6 Hypervisor in Cloud Computing

Virtualization involves a shift in thinking from physical resource to logical, improves IT resource utilization by treating your physical resources as pool of resources where virtual resources will be dynamical allotted. By implementing virtualization in your work environment, you're able to consolidate resources like processors, storage, and network into a virtual environment. System virtualizaiton creates several virtual systems inside one physical system; virtual systems unit are not dependent on operating environment that use virtual resources. System virtualization is most typically enforced with hypervisor technology; hypervisor or virtual machine

manager are firmware or software components that are capable of virtualizing system resources, and managing them [13].
Typically hypervisors are classified into two categories

- Type 1 Hypervisor
- Type 2 Hypervisor

2.6.1 Type 1 Hypervisor:

This sort of hypervisor is deployed as a bare-metal installation. This suggests that the primary thing is to install the hypervisor as the operating system on the server. The good thing about this is that the hypervisor can communicate directly with

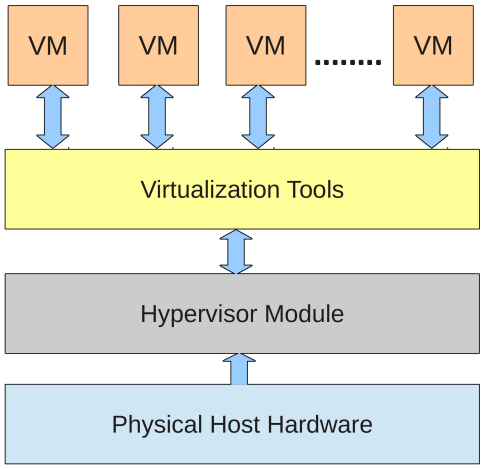

Figure 2.2: Type 1 Hypervisor

the underlying physical server hardware. Those resources are then virtualized and delivered to the running VMs. This is often the well-liked methodology for several production systems [14].

2.6.2 Type 2 Hypervisor:

This model is additionally referred to as a hosted hypervisor. The software isn't put in onto the bare-metal, however instead is loaded on top of already installed live operating system. Though there's an additional hop for the resources to take after they experience to the VM the latency is minimal and with todays fashionable software enhancements, the hypervisor will still perform optimally [14].

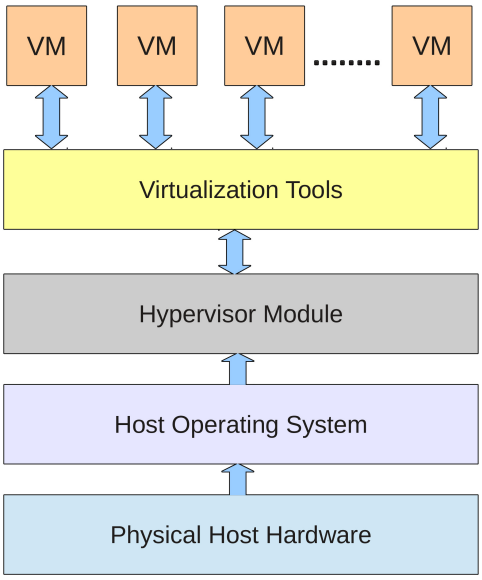

Figure 2.3: Type 2 Hypervisor

Hypervisors are based on two types [15] of virtualization:

- Full Virtualization.

- Para Virtualization.

2.6.3 Full Virtulization:

Most hypervisors typically uses full virtulization which suggests that they utterly emulate all hardware devices to the machines. Guest operting systems don't need any modification and behave as if they each have exclusive access to the whole system.

Full virtulization typically includes performance drawbacks as a result of complete emulation typically demands a lot of process resources (and a lot of overhead) from the hypervisor.

2.6.4 Para Virtualization:

Paravirtualization is a virtualization technique that presents a software system interface to the virtual machines that's almost like but not same as that of the underlying hardware. The intent of this changed interface is to cut back the portion of the guest operational system's execution time that's spent performing operations that are substantially tougher to run in an exceedingly virtual surroundings compared to a non virtualized environment surroundings. There are specifically defined "hooks" that enable the guest and host to request and acknowledge these tough tasks that will rather be executed within the virtual domain, where execution performance is slower.

2.7 Open-source Cloud Computing Solutions

The event of cloud computing solutions brings many technical challenges to cloud developers. These challenges is sorted in three main areas: negotiation, decision, and operation [12]. Within the negotiation phase, there are the challenges relative to application developers interface with the cloud additionally because the description of the cloud offerings. It includes the definition of the programmability level that the cloud solution can supply. The decision phase copes with most drawbacks that clouds faces behind the scenes; however virtual resources is scheduled to fulfill user needs. Last, the operation phase is related to the social control of selections and also the communication between cloud components.

2.7.1 Xen Cloud Platform (XPC)

The Xen hypervisor could be a answer for infrastructure virtualization that gives an abstraction layer between servers hardware and the software [16] [17]. Xen is associate open supply type-1 or bare metal hypervisor, that makes it doable to run several instances of softwares or so completely different operative systems in parallel on one machine. Xen is employed because the basis for variety of industrial and open-source applications, such as: server virtualization, Infrastructure as a Service (IaaS), desktop virtualization, security applications, embedded and hardware appliance. Xen allows user to extend server utilization, consolidate server farms, scale back quality, and reduce total value of possession. The Xen is employed by several cloud solutions like Amazon EC2, Nimbus and Eucalyptus.

2.7.2 Nimbus

Nimbus [16] [18] is open-source solution to deploy clusters into Infrastructure as a Service for cloud computing focusing principally on scientific applications. It offers to users the likelihood to apportion and assemble remote resource by deploying virtual machines, referred to as virtual space services. A virtual space service is a virtual machine manager. To deploy applications, Nimbus offers a cloudkit configuration that consists of service hosting and image repository. Combining all the tools and capabilities in numerous ways in which permits users to quickly develop custom community specific solutions. For instance, one community would possibly need to use Nimbus IaaS to assemble a private of community cloud whereas the other might want to concentrate on augmenting resources of an already exiting cloud with resources from varied community and public clouds.

2.7.3 OpenNebula

OpenNebula is an open-source toolkit accustomed build personal public and hybrid clouds [19] [16] [20]. It's been designed to be integrated with networking and

storage solutions and to suit into existing data-centers. The target is to develop the most-advanced, highly-scalable and adoptable solutions for building and managing virtualized data centers and IaaS clouds. It give cloud developers and users with alternative of cloud and system interfaces type, open cloud to de-facto standards, to supports the bound of an expensive scheme of upper level parts. They arrange for operation network to alter the management of OpenNebula cloud instances, fault tolerance practicality to maximise time period within the cloud, increased management of images and templates, new security practicality, increased support for federation of data-centers and support for multi-tier architectures.

2.7.4 Eucalyptus

Eucalyptus is an open-source cloud computing framework targeted on educational analysis. Eucalyptus users are ready to begin, control, access and terminate entire virtual machines [16] [21]. Eucalyptus project presents four characteristics that differentiate it from others cloud computing solutions: (a) Eucalyptus was designed to be simple while not requiring dedicated resources; (b) Eucalyptus was designed to encourage third-party extensions through standard software and language-agnostic communications mechanisms; (c) Eucalyptus external interface is based on the Amazon API and (d) Eucalyptus provides a virtual network overlay that isolates network traffic of various uses and permits clusters to be a part of a similar native network.

2.8 Conclusion

In this chapter an overview of the cloud computing is given, in which different deployment models, service models have been described. A differentiation between virtualization and cloud computing is also presented along with virtualization technologies and open source cloud computing solutions.

Chapter 3

Virtual Machine Placement Strategies in Cloud Computing

Categorization of VM placement algorithms

VM Placement Based on Constraint Programming

VM Placement Based on Stochastic Integer Programming

VM Placement Based on Bin packing Approch

VM Placement Based on Genetic Algorithm

Strategies Followed by Open-Source Solutions

Chapter 3

Virtual Machine Placement Strategies in Cloud computing

3.1 Introduction

Virtual machine placement is the method of mapping virtual machines to physical machines. In different words, virtual machine placement is the method of choosing the foremost appropriate host for the virtual machine. The method involves categorizing the virtual machines hardware and resources necessities and the anticipated usage of resources and therefore the placement goal. The primary goal may either be increasing the usage of accessible resources or it will be saving of power by having the ability to stop working some servers. The involuntary virtual machine placement algorithms are designed keeping in mind the on top of goals.
The chapter is organized as follows. In section 3.2 categories of VM placement algorithm is discussed. In section 3.3 VM placement based on constraint programming is discussed. In section 3.4 VM placement based on stochastic integer programming is discussed. In section 3.5 VM placement based on bin packing is discussed. In section 3.6 VM placement based on genetic algorithm is discussed. In section 3.7 strategies followed by open-source solutions are discussed.

3.2 Categorization of VM placement algorithms

The placement algorithms may be loosely classified into 2 classes on the basis of their placement goal.

- Power primarily based approach
- Application QOS primarily based approach

3.2.1 Power primarily based approach

The necessity of power management has become progressively evident in computing environments. The necessity for power management is driven by 2 factors:

1. The increasing demands on power by each computing and cooling resources during operation of a data center.
2. The rising price of power.

The main aim of those approaches is to map virtual machines to physical machines in such the way, in order that the servers may be utilized to their most potency, and therefore the different servers may be either hibernated or clean up looking on load conditions.

3.2.2 Application QOS primarily based approach

These algorithms manage the mapping of virtual machines onto physical hosts with the aim of maximizing the standard of service (QOS) delivered. By endlessly observance virtual machine activity and using advanced policies for dynamic workload placement, such algorithms will cause higher utilization of resources eventually resulting in savings in price.

3.2.3 Dynamic and static approaches

The placement computations may be either dynamic or static. Static algorithms usually use the information that's antecedently collected as input. After the initial calculations, the mapping might not be recomputed for long period, such as many months. The computations are largely done off-line. In distinction, dynamic allocation is enforced on shorter time scales, ideally shorter than periods of significant variability of the resource demand. The placement algorithms run within the background of the application processes assembling information [22].

3.3 VM Placement Based on Constraint Programming

One of the aims of cloud suppliers is to modify management of virtual machines taking the standard of service necessities of applications into thought. This problem may be developed as a constraint programming problem. The goal of the constraint programming is to maximize a global utility function. This global utility function is chosen to take SLA fulfillment and in operation prices into thought. The utility functions mentioned within the paper [23] maps the present state of every application (workload, resource capability, SLA) to scalar price. This scalar price tries to quantify the applications satisfaction with relation to the goals that are set by the automated manager.

The virtual machine placement employing a constraint based approach may be thought of as a two-stage method.

1. Local Decision - it's related to every application environment.

2. Global Decision - It takes as input the local decision from all applications, so tries to maximize the global utility function.

3.3.1 VM placement as constraint satisfaction problem

Any application hosted within the cloud is referred as application environment(AE). Typically one virtual machine (VM) is related to just one AE. A module called as local decision module is related to every AE that computes the associate utility function. Given with current workload, this utility function offers us a live measure of the application satisfaction with a particular resource allocation. This output is given to a module called global decision module that really maps the VMs to Physical machines (PMs) taking into consideration the mainframe load of virtual and physical machines.

Formalising the problem

- $AE = \{ae_1, ae_2, ae_3, ..., ae_i, ..., ae_m\}$ denotes set of AEs.
- $H = \{h_1, h_2, h_3, ...h_i..., h_q\}$ denotes set of physical machines (PMs).
- There are n classes of virtual machines (VM) available in the set $V = \{v_1, v_2, ..., v_k, ..., v_n\}$ where $v_k = (v_k^{cpu}, v_k^{ram})$ specifies the cpu and memory of the virtual machine [23].

Local decision module

These decision unit related to a selected application environement. It takes under consideration a hard and fast service-level utility function that maps service level to a utility value and a dynamic resource-level utility function that maps a resource capacity to a utility value. The results of resource-level utility function mapping is communicated to global decision Module in each iteration. For each application environment ae_i the resource-level utility function U_i is defined as $U_i = f_i(L_i)$ where L_i is the virtual machine allocation list for application environment ae_i.

$$L_i = (n_{i1}, n_{i2}, n_{i3}, ..., n_{ik}, ...n_{im})$$

where n_{ik} is the number of virtual machines of class v_k associated with application ae_i. For each application their is an associated upper bound for the number of

virtual machine of each class ($L_i^{max} = (n_{i1}^{max}, n_{i2}^{max}, ..., n_{ik}^{max}, ..., n_{im}^{max})$) and the total number of virtual machines (T_i^{max}) it is ready to accept. The constraints of these application are delineated below:

Constraints:

1. $n_{ik} \leq n_{ik}^{max}$, $1 \leq i \leq m$ & $1 \leq k \leq n$
2. $\sum_{i=1}^{n} n_{ik} \leq T_i^{max}, 1 \leq i \leq m$

Global decision module

The global decision module involves two phases

1. Virtual machine provisioning
2. Virtual machine Placement

The above two phase can be represented as constraint solving problem as delineated below.

Virtual machine provisioning phase:

This phase aims to find the virtual machine allocation list for each application environment ae_i, and simultaneously tries to maximize the global utility value U_{global}. The VMs allotted to all or any applications are certain by the constraints on the full capacity of physical servers.

$$\sum_{i=1}^{m} \sum_{k=1}^{n} n_{ik}.v_k^{cpu} \leq \sum_{j=1}^{q} C_j^{cpu}$$
$$\sum_{i=1}^{m} \sum_{k=1}^{n} n_{ik}.v_k^{ram} \leq \sum_{j=1}^{q} C_j^{ram}$$

where C_j^{cpu} and C_j^{ram} are CPU and RAM capacity of physical host h_j.

The global utility function is represented as:

$$U_{global} = maximize \sum_{i=1}^{m} (\alpha_i \times u_i - \epsilon.cost(L_i))$$

where $0 < \alpha_i < 1$ and $\sum_{i=1}^{m} \alpha_i = 1$

ϵ is a constant that determines the tradeoff between fulfillment of performance goal as critical value of the desired resources. $cost(L_i)$ is a function of virtual machine allocation list. This function's output is L_i that is a list represents the allocation of

virtual machines to physical host machines.

Virtual machine placement phase:

This phase takes the virtual machine allocation list L_i as it's input from the virtual machine provisioning phase. In this phase, global decision method creates a single list VL from all L_i's. This list contains all the virtual machines running in the current time.

$$VL = \{vm_1, vm_2, vm_3, ..., vm_l, ..., vm_v\}$$

For every physical host $h_j \in H$, the bit list represents $B_j = \{b_{j1}, b_{j2}, b_{j3}, ..., b_{jl}, ..., b_{jv}\}$ represents the set of virtual machine assigned to h_j. Let $R = \{r_1, r_2, r_3, ..., r_l, ..., r_v\}$ represents the resource capacity of all virtual machines. The constraints can be express as:

$$\sum_{l=1}^{v} r_l^{cpu}.b_{jl} \leq C_j^{cpu}, 1 \leq j \leq q$$
$$\sum_{l=1}^{v} r_l^{ram}.b_{jl} \leq C_j^{ram}, 1 \leq j \leq q$$

This phase aims to minimize the active number of physical machines A. The function is

$$\sum_{j=1}^{q} u_j \quad ,$$

where $u_j = 1$ if $\exists vm_l \in VM | b_{jl} = 1$, otherwise it is 0.

The global decision function runs periodically, and calculates the difference between virtual machine placement produced at the finish of current step and with the preceding step, this will give the virtual machine migration information to the virtual machine manager.

3.4 VM Placement Based on Stochastic Integer Programming

Stochastic integer programming is beneficial in cases where actual demands don't seem to be framed however the distribution of demands is understood or is calculable. Most of the cloud suppliers supply two payment plans to users for resource provisioning.

These are:

1. Reservation plan
2. On-Demand plan

With the assistance of SIP (Stochastic integer Programming), the longer term demands that aren't sure are often taken into thought, and along side this the distinction within the costs of reservation and on-demand plans may be thought about. Taking under consideration all of this, a mapping of virtual machines to physical machines is generated thus on minimize the price spent in every arrange for hosting virtual machines in multiple cloud supplier environment minimizing the issues faced by under provisioning or over provisioning. The trade-off factor is that the On-demand value versus oversubscribing value.

An algorithm delineated in [24] that uses Stochastic integer programming. It allocates resources in three phases.

- Reservation : Cloud broker provisions resources while not knowing the demand of users.
- Utilization : It uses the reserved resources.
- On-demand : Just in case the demand exceeds the reserved resources, the extra resources can be requested in case of on-demand pay setup.

Each phase is associated with a certain cost.

3.4.1 VM placement based on SIP

In the cloud computing environment, we've variety of cloud suppliers that supply pool of resources to the user. The virtual machines may be divided into a collection of categories with every category representing a special application type. The target is to attenuate all the prices, at an equivalent time meeting the demand of users. The demands and costs aren't fixed. However, we can estimate the demand and costs

supported the distribution curve. Thus, SIP would be helpful to find a solution of VM placement problem. The problem statement may be outlined formally as:

- $\nu = \{v_1, v_2, ..., v_{last}\}$ denotes the set of VM categories. A VM category represents the application type.

- $\rho = \{p_1, p_2, ..., p_{last}\}$ denotes set of cloud suppliers. Every cloud supplier provides a pool of resources to user.
  The superscripts $(h), (s), (n)$ and (e) correspond to the four resources provided by cloud suppliers, particularly computing power, storage, network bandwidth and electric power respectively.

- $t_j^{(h)}, t_j^{(s)}, t_j^{(n)}$ represents to the utmost capability of corresponding resource which cloud supplier p_j will offer to user.

- $r_j^{(h)}, r_j^{(s)}, r_j^{(n)}$ represents amount of corresponding resource quantity by one VM beneath category v_i .

- $c_j^{(h)}, c_j^{(s)}, c_j^{(n)}$ represents the costs of corresponding resources in reservation phase for cloud supplier p_j.

- $\bar{c}_j^{(h)}, \bar{c}_j^{(s)}, \bar{c}_j^{(n)}$ represents the costs of corresponding resources in utilization phase for cloud supplier p_j.

- The cost of resources in on-demand part could be random. $\tilde{c}_j^{(h)}, \tilde{c}_j^{(s)}, \tilde{c}_j^{(n)}$ denotes the costs of corresponding resources in on-demand part for cloud supplier p_j.

- Uncertainty of prices and Demands.

  - $D_i = \{d_{i1}, d_{i2}, ..., d_{iv}\}$ denotes set of maximum variety of needed VMs of category V_i . The full variety of needed VMs, D are cartesian product of all D_i over all i.
  - $\tilde{C}_j^{(h)}, \tilde{C}_j^{(s)}, \tilde{C}_j^{(n)}$ represents set of possible costs of offered resources by supplier p_j in on-demand phase.

- $\upsilon_i(d)$ denotes the amount of needed VMs in the category V_i if demand d is realised.

Phase I This defines the amount of VMs to be provisioned in reservation section.
Phase II This defines the amount of VMs allotted in utilization and on-demand section.

The SIP formulation can be represented as

$$\sum_{v_i \in \nu} \sum_{p_j \in P} c_{ij} X_{ij}^r + \xi \Omega[\wp(X_{ij}^r, \omega)]$$

where

X_{ij}^r represents number of virtual machine provisioned in phase I.

$\xi \Omega[\wp(X_{ij}^r, \omega)]$ represents the cost in phase II. Here $\omega \in \Omega = D \times \prod_{p_j \in P}$ represents set of price and demands.

The objective is to maximize this function

Taking the probabilities of prices and demands being realised, we can expand the on top of equation as follows.

$\sum_{v_i \in \nu} \sum_{p_j \in P} c_{ij} X_{ij}^{(r)} + \sum_{v_i \in \nu} \sum_{p_j \in P} \sum_{m \in \bar{C}_j} \sum_{d \in D} p_j(m) p(d) (\bar{c} X_{ij}^{(r)}(m,d) + \tilde{C}(m) X_{ij}^{(u)}(m,d) + \tilde{C}(m) X_{ij}^{(o)}(m,d))$

Subjected to constraints

$$X_{ij}^{(u)}(m,d) \leq X_{ij}^{(r)}, v_i \in \nu, p_j \in P, m \in \bar{C}_j, d \in D$$
$$\sum_{p_j \in P} X_{ij}^{(u)}(m,d) + X_{ij}^{(o)}(m,d) \geq \upsilon_j(d), v_i \in \nu, m \in \bar{C}_j, d \in D$$
$$\sum_{v_i \in \upsilon} r_i^{(h)} X_{ij}^{(u)}(m,d) + X_{ij}^{(o)}(m,d) \leq t_j^{(h)}, p_j \in P, m \in \bar{C}_j, d \in D$$
$$\sum_{v_i \in \upsilon} r_i^{(s)} X_{ij}^{(u)}(m,d) + X_{ij}^{(o)}(m,d) \leq t_j^{(s)}, p_j \in P, m \in \bar{C}_j, d \in D$$
$$\sum_{v_i \in \upsilon} r_i^{(n)} X_{ij}^{(u)}(m,d) + X_{ij}^{(o)}(m,d) \leq t_j^{(n)}, p_j \in P, m \in \bar{C}_j, d \in D$$
$$X_{ij}^{(r)}(m,d), X_{ij}^{(u)}(m,d), X_{ij}^{(o)}(m,d) \in 0, 1, ..., v_i \in \upsilon, p_j \in P, m \in \bar{C}_j, d \in D$$

3.5 VM Placement Based on Bin packing Approch

The bin packing approach can be used to realize the particular mapping of virtual machines to available physical machines. It's potential to attenuate the value of running the data-center by tightly packing the VMs needed to be running at a time onto the smallest amount of physical machines possible.

3.5.1 VM placement as Bin packing problem

The Physical machines can be thought about as bins and the virtual machines to be placed can be thought-about as objects to be placed within the bins. A 3-stage algorithm is given as [22].

- Formulate a pattern of past demands.
- Forecast the longer term demand supported the pattern of past demands.
- Map or Remap virtual machines to physical machines, this can be called Measure-Forecast-Remap(MFR).

The algorithm aims at minimizing the quantity of hosts needed for virtual machines. The last stage of the algirithm uses bin packing. The demand within the next interval is foreseen on the premise of demand within the previous interval. Supported these a heuristic is developed. Supported the solution to the present heuristic, the virtual machines are packed onto the physical machines. The [22] paper uses first fit approximation.

Formalising the problem

- Given N virtual machines and M physical hosts each having a capacity C^m.
- R is denoted as remapping interval.

- The resource demand of n^{th} VM at begining of interval i is represented as U_i^n.
- The distribution of demand corresponding to the time series U_i^n is $u^n(x)$.
- $f_{i,k}^n$ Represents the forecast demand k units of time prior to current time i for virtual machine n.
- The capacity required to ensure an error rate less than p is denoted by $c_p(\mu, \sigma^2)$where μ and σ^2 is mean and variance of prediction error distribution.

Based on the forecast of demand of every virtual machines, the virtual machines are sorted in descending order. Every virtual machine is then embarked on the list and a trial is formed to position it on the first physical machines wherever it fits, *i.e.*, after its placement, the summation of resource demands of all virtual machines on that of physical machines doesn't exceed the whole capacity of the physical machines. The first-fit algorithm minimizes the quantity of active physical machines.

3.6 VM Placement Based on Genetic Algorithm

Genetic algorithm is a heuristic based on search technique. It's notably helpful in issues where objective functions dynamically varies. Genetic algorithm typically starts with a population of solution, applies genetic operators on this that finally ends up providing best solution [25].

A variation to genetic algorithm called Grouping Genetic algorithm can even be applied to the virtual machine Placement problem. These algorithms will take under consideration extra constraints whereas optimizing the cost function. This is often notably helpful in cases where we want to control on groups [26].
Genetic algorithm mainly involves:

- Chromosome modeling - This is often associate encoding schema used for encoding the details of the problem that's to be passed from one generation to the other.

- Population data formatting - The feasibility of the solution relies on the feasibility of the initial solution.
- Crossover - It's a genetic operator used for varying the programming of the chromosome from one generation to next generation.
- Mutation It is a genetic operator that alters one or more additional gene values of a chromosome from its initial state. This helps to forestall the population from stagnating at any local optima.
- Generation Alternation - This is often where we tend to choose one solution from the next generation of solution. [25]

3.6.1 VM placement as grouping GA problem

The Grouping Genetic algorithmic program is thought of as a bin packing problem where the aim is to not solely realize a solution with highest packing potency however additionally to satisfy the constraints. The algorithm in [26] offers a mechanism to take into account VM interference.

Formalising the problem

- Y_i a binary variable whose value is 1 if the i^{th} server is used, otherwise 0.
- X_{ij} a binary variable whose value is 1 if the j^{th} server is consolidated into i^{th} target server, otherwise 0.
- $attr_{jk}$ represents k^{th} attribute usages for j^{th} server.
- A normalization parameter K is used for setting the priority between minimizing amount of bins versus packing efficiency maximization.
- i and $\bar{i}$ are used as new server index.
- j and $\bar{j}$ are used for application index.

- J_a is a subset of applications which can not be placed on the same target server.
- k is used as server attribute index.

The objective is to minimize

$$Obj = \sum_i Y_i - K * \sqrt{\sum_i (\sum_k (\frac{\sum_j attr_{jk} * X_{ij}}{Attr_{ik}})^2)}$$

Subjected to constraints

$$\begin{aligned}
&\sum_i X_{ij} = 1, && \forall j \\
&Y_i \geq X_{ij}, && \forall i, j \\
&\sum_j X_{ij} \geq Y_i && \forall i \\
&Y_i * Attr_{ik} \geq \sum_j X_{ij} * attr_{jk} && \forall i, j, k \\
&\sum_{j \in J_a} X_{ij} \leq 1 && \forall i, a \\
&X_{\bar{i}\bar{j}} = 0 && \exists \bar{i}, \bar{j}
\end{aligned}$$

3.7 Strategies Followed by Open-Source Solutions

To decide this allocation various cloud computing model use different algorithms [27] as shown in table 3.1.

3.7.1 Rank Scheduling Algorithm

The virtual machine scheduler used in OpenNebula is known as match making scheduler, which uses a predefined rank [4] [27] for prioritizing the available resources. Based on the priority, this scheduler places the virtual machine on the host having the highest priority . This algorithm takes requirements and predefined rank as its input and produces the resource number in which to place the virtual machine as output. When a virtual machine request arrives at the scheduler, first it sweeps away the resources which are not befitting into the requirement. The detail steps for Rank placement policy is delineated in Algorithm 1.

Algorithm 1 Rank Scheduling Algorithm

1: **procedure** ALLOCATEVM(VM_{List})
2: $i \leftarrow 1$;
3: **while** $VM_{List} \neq \phi$ **do**
4: $HostId \leftarrow RankingAlgo(Host_{List}, Requirment_i, Rank)$;
5: $Host_{HostId} \leftarrow VM_i$;
6: $i \leftarrow i + 1$;
7: **end while**
8: **end procedure**
1: **procedure** RANKINGALGO($Host_{List}, Requirment, Rank$)
2: **for** each $Host \in Host_{List}$ **do**
3: **if** Host Satisfies the Requirments **then**
4: $add(Member_{List}, Host)$;
5: **end if**
6: **end for**
7: $Priority_{List} \leftarrow SortByRank(Member_{List}, Rank)$;
8: $Allocated_{Host} \leftarrow Priority_{List}(1)$;
9: $Return\ Allocated_{Host}$;
10: **end procedure**

Table 3.1: VM placement approch by different tools

Tools	Provisioning model	Default placement policies	Configurable placement policies
Nimbus	Immediate	Static-greedy and round-robin resource selection	No
Eucalyptus	Immediate	Static-greedy and round-robin resource selection	No
OpenNebula	Best-effort	Initial placement based on requirement/rank policies to prioritize those resources more suitable for the virtual machine (VM) using dynamic information and dynamic placement to consolidate servers	Support for any static/dynamic placement policy

3.7.2 Greedy First Fit Algorithm

For placement of a virtual machine on the underlying host Eucalyptus and Nimbus uses Greedy First fit algorithm [27]. In First fit Greedy strategy whichever node that can run the virtual machine found first is selected as the host for virtual machine placement [28]. This algorithms takes virtual machine request and requirment as input and produce the resource number in which to place the resource as output. The detail steps for Greedy first fit placement policy is delineated in Algorithm 2 .

3.7.3 Round-Robin Algorithm

For placement of a virtual machine on the underlying host Eucalyptus and Nimbus also use Round-Robin algorithm [27]. Round robin records the last position of the scheduler visited. And also the scheduler starts from the last visited position next time new request(s) come(s) meantime the resources are thought of as a circular linked list [28]. This algorithms takes virtual machine request and requirment as

Algorithm 2 Greedy First Fit Algorithm

procedure ALLOCATEVM(VM_{List})
    $i \leftarrow 1$;
    while $VM_{List} \neq \phi$ **do**
        $HostId \leftarrow GreedyAlgo(Host_{List}, Requirment_i)$;
        $Host_{HostId} \leftarrow VM_i$;
        $i \leftarrow i + 1$;
    end while
end procedure

procedure GREEDYALGO($Host_{List}, Requirment$)
    for $i = 1$ to n **do** ▷ n is the number of physical host
        if $Host_i$ Satisfies the Requirments **then**
            Rerurn $Host_i$;
        else
            Continue;
        end if
    end for
end procedure

Algorithm 3 Round-Robin Algorithm

```
1: procedure AllocateVM(VM_List)
2:     i←1;
3:     pos←1;
4:     while VM_List ≠ φ do
5:         HostId ←RoundRobinAlgo(Host_List, Requirment_i, pos);
6:         Host_HostId←VM_i;
7:         i ←i + 1;
8:     end while
9: end procedure
1: procedure RoundRobinAlgo(Host_List, Requirment, pos)
2:     end ←pos;
3:     limit ←n + 1;                          ▷ n is the number of physical host
4:     if pos > n then
5:         pos ←1;
6:     end if
7:     for i = pos to limit do
8:         if i <= n & Host_i Satisfies the Requirments then
9:             Rerurn Host_i;
10:        else
11:            if i > n then
12:                pos ←1;
13:                limit ←end;
14:            else
15:                Continue;
16:            end if
17:        end if
18:    end for
19: end procedure
```

input and produce the resource number in which to place the resource as output. The detail steps for Roudn-Robin placement policy is delineated in Algorithm 3.

In the above scheduling approaches, Greedy and round robin that provided by Eucalyptus and Nimbus is a random technique to pick adaptive physical resources for the VM requests that not considering maximum usage of physical resource.

3.8 Conclusion

In this chapter different VM placement strategies have been discussed along with the strategies followed by the open source technologies. Different strategies discussed include VM placement based on constraint programming, stochastic integer programming, bin packing, genetic algorithm, ranking, greedy first fit, round-robin.

Chapter 4

Proposed Scheme

Framework

VM Scheduler

Time Complexity Analysis

Chapter 4

Proposed Scheme

4.1 Introduction

The chapter 4 is organized as follows. In section 4.2 proposed framework is discussed. In section 4.3 proposed VM Scheduler is discussed along with all the algorithms. In section 4.4 the time complexity analysis of the proposed schemen is given.

4.2 Framework

Figure 4.1 elucidates the model for allocation of virtual machines to physical host machines present in the datacenters. Client/user can place their list of requirements for any service in the interface provided with the help of a web browser, that may require computational and storage infrastructure. That request is communicated to the cloud service provider through Internet. The service provider will serve the requests by placing individual requests to suitable virtual machines that can perform the requisite operations. The virtual machine specification is now forwarded to the hypervisor of the service provider. Hypervisor uses the virtualization tools to create virtual machines of the specified specification. The virtual machine manager keeps track of the created virtual machines. But the problem is to map the virtual machine requests to the host machines.

To resolve this problem virtual machine manager sends the virtual machine specifications to the virtual machine scheduler (VM Scheduler). But in this model instead of sending that request directly, a binary search tree of the requested virtual machines is created and that is sent to the VM Scheduler. This is done to improve the resource utilization, which is explained in the next sub section in the algorithm. Now VM scheduler will take the maximum requirement node from that virtual machine tree and will search for a host that will best fit the requirement. For implementing the best-fit strategy all the physical resources are also logically organized in a binary search tree, since searching a host using this data structure will take in average $O(logN)$ time where N is the number of physical hosts.

This will reduce the time for allocation. After getting the proper host the scheduler will return the host number to the virtual machine manager for placement of virtual machine on that host. Now the virtual machine manager has all information about the virtual machine and its location; it will now send a service activation message to the client/user. After that client/user can access the service for the duration specified.

4.3 VM Scheduler

Let $Host_{Tree} = \{H_1, H_2, H_3, ..., H_n\}$ is the list of physical hosts arranged in a tree, available which have to process the virtual machine requests. The virtual machine requests that have been made in a particular time interval are collected at the virtual machine manager and are stored in a VM_{Tree}, $VM_{Tree} = \{VM_1, VM_2, VM_3, ..., VM_n\}$. The time interval is used to avoid the problem of starvation and is very small to avoid the delay. Here we have to do a mapping $VM_{Tree} \rightarrow Host_{Tree}$, the mapping should be done in such a way that it will optimize the use of resources and should be faster. Figure 4.2 elucidate the flow chart of the proposed scheme where best fit is decided by Algorithm 5 which uses the following fracion value.

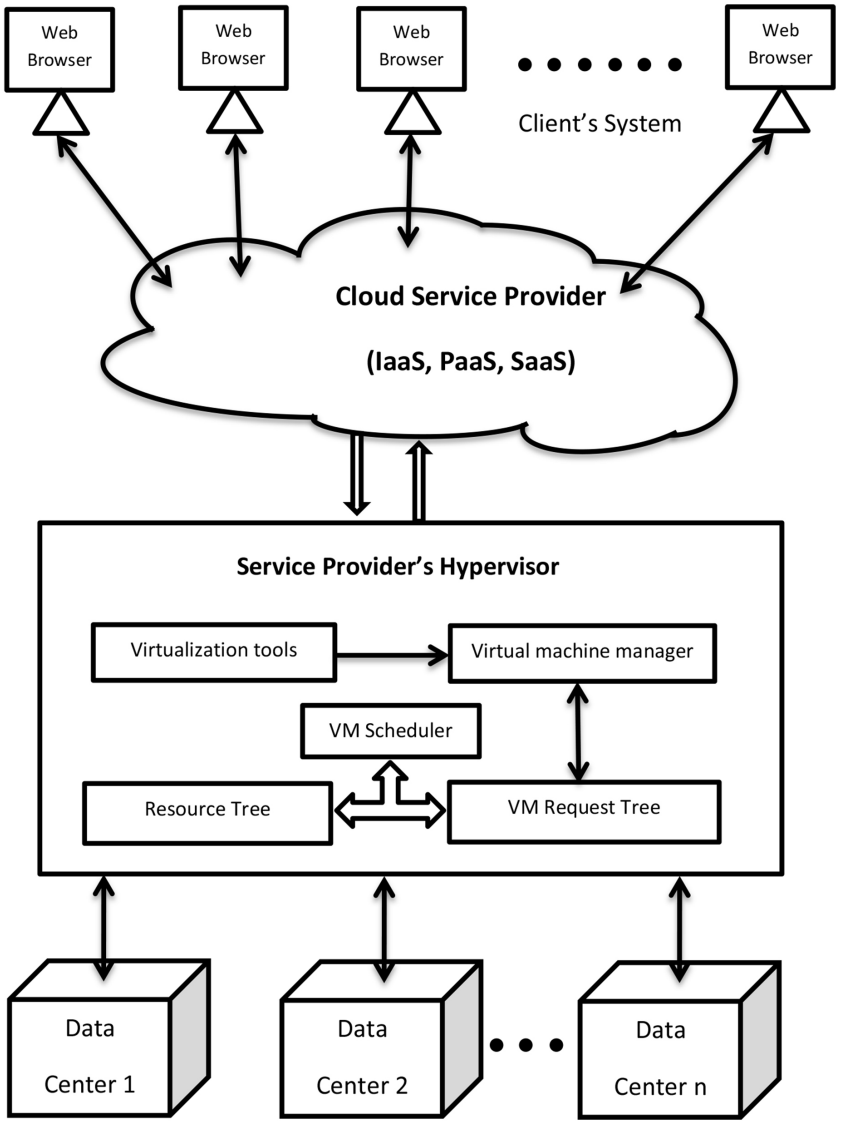

Figure 4.1: Proposed Framework

$$\frac{RVMS}{AHS} = \begin{cases} 1 & \text{Perfect fit.} \\ (0,1) & \text{Possible candidate.} \\ (1,+\infty) & \text{Reject.} \end{cases}$$

Where

$RVMS$ is requested virtual machine specification.

AHS is the available host specification.

When a virtual machine request arrives at the scheduler, it calcutes this value at each host arranged in the binary search tree starting from the root node traversing towards the leaf node, until it finds a best fit. If it finds this value to 1 for a perticular host then that is the perfect fit and that host nood is assigned as the best fit node for that virtual machine. If the value calculated is greater than 1 then that host is not a candidate host for that virtual machine and the search will continues to its right child host node. If the value is less than 1 i.e. a fraction between 0 to 1 then the search bigins in the direction of left host node and the value at each left and right child node is calculated. If the value is found less than the value that has been calculate at its parent host node then it will stop searching further and declare its parent node as the best fit node.

For example let there are five host nodes H_1 to H_5 present at the datacenter having memory specification (in MB) 1024, 360, 525, 575, 680 respectively. These host machines are organized in the host tree as shown in the figure 4.3.

Suppose a virtual machine request having memory specifacation 480 MB arrives at the scheduler. Now scheduler will calculate the $\frac{RVMS}{AHS}$ value at the root node $i.e.$ H_5. The value is found to be 0.71. Now the search will continue in left direction $i.e.$ H_3. The value at H_3 is 0.91. Next host node is H_2 and the value is calculated to 1.33 which is greater than 1, hence rejected. So the best fit host is its parent host node having the calculated value 0.91. Hence host H_3 is the best host to run this virtual machine request. The purpose of arranging the virtual machine requests in

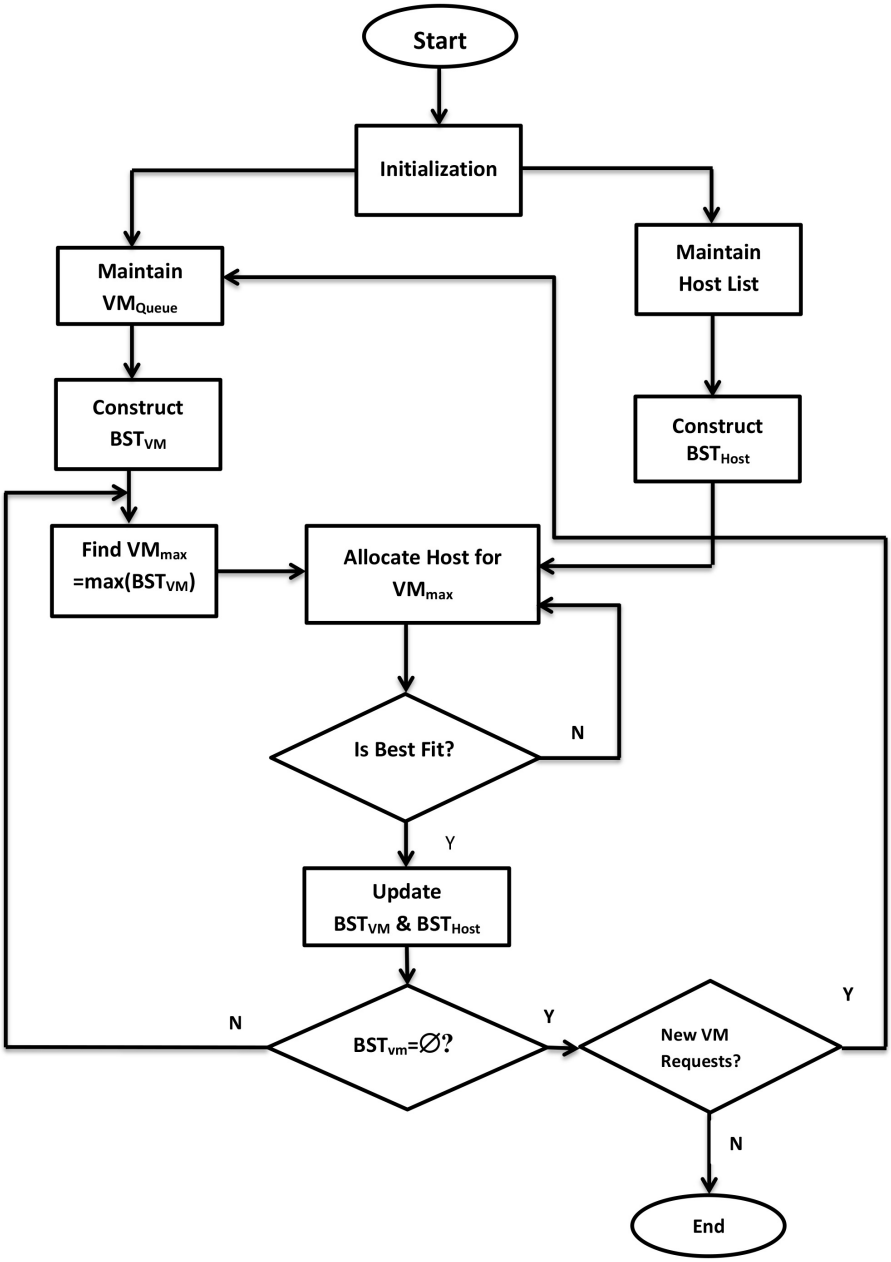

Figure 4.2: Flow chart for the Proposed Scheme

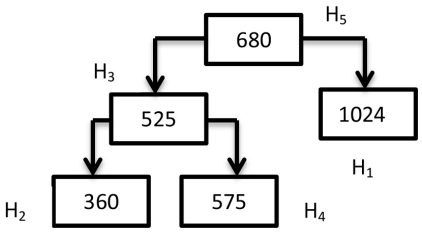

Figure 4.3: Example of host representation

binary search tree is: suppose two virtual machine requests arrive at a time having memory requirment 510 MB and 400 MB. If the allocation of 510 MB VM is processed first then it will get allocated at the H_3 host, and 400 MB request is allocated at H_4 host. The remaining memory at H_3 is 15 MB and at H_4 175 MB. But if the 400 MB request is processed first then first it will get allocated to host H_3, the remaining memory is 125 MB and 510 MB request is allocated to host H_4, the remaining memory at H_4 is 65 MB. So in the first case we are left with big fragment of memory *i.e.* 175 MB, which can be used to allocate some new requests. In the second case we got two fragments of memory but that may not get utilized (lets say when a VM request for 150 MB memory arrives). Hence this technique improves the resource utilization.

4.3.1 VM Scheduler Algorithm

Algorithm 4 VM Scheduler Algorithm

1: **procedure** VMSCHEDULER(BST_{VM}, BST_{Host})
2: **while** $BST_{VM} \neq \phi$ **do**
3: $currTh \leftarrow 0$;
4: $VM^{Max} \leftarrow BST_{VM}^{Max}$;
5: $Max \leftarrow BST_{Host}^{Max}$;
6: **if** $VM^{Max} > Max$ **then**
7: $exit(0)$;
8: **else**
9: $Host_{Alloc} \leftarrow Allocate(VM^{Max}, BST_{Host}, currTh)$;
10: **end if**
11: $Delete(BST_{VM}, VM^{Max})$;
12: $Update(Host_{Alloc}, VM^{Max})$;
13: **end while**
14: **end procedure**

Algorithm 5 Allocate Algorithm

1: **procedure** ALLOCATE($VM^{Max}, BST_{Host}, currTh$)
2: $threshold \leftarrow \frac{VM^{Max}}{BST^{Root}_{Host}}$;
3: **if** $threshold == 1$ **then**
4: $return\ BST^{Root}_{Host}$;
5: **end if**
6: **if** $threshold < 1$ & $currTh < threshold$ **then**
7: $currTh \leftarrow threshold$;
8: $Allocate(VM^{Max}, BST^{Left}_{Host}, currTh)$;
9: **else**
10: **if** $threshold > 1$ & $currTh < threshold$ **then**
11: $currTh \leftarrow 0$;
12: $Allocate(VM^{Max}, BST^{Right}_{Host}, currTh)$;
13: **else**
14: Wait for free Host;
15: **end if**
16: **end if**
17: $return\ BST^{Root}_{Host}$;
18: **end procedure**

Algorithm 6 Update Algorithm

1: **procedure** UPDATE($Host_{Alloc}, VM^{Max}$)
2: $Host^{new} \leftarrow BST^{Host_{Alloc}}_{Host} - VM^{Max}$;
3: $Delete(BST_{Host}, BST^{Host_{Alloc}}_{Host})$;
4: $Insert(BST_{Host}, Host^{new})$;
5: **end procedure**

4.4 Time Complexity Analysis

Allocate Procedure:

In average case let $T(n)$ = Number of comparison needed to search the best fit among n hosts. Each time we call the Allocate procedure in $step-8$ or $step-12$ we remove half of element form the serch space. So we will be just searching in $T(\frac{n}{2})$. $step-2$ and $step-3$ will be executed once each time the procedure is called, hence $C_1.1$ and $C_2.1$ respectively. If $step-6$ will be evaluated to true then $step-6$ and $step-7$ will be evaluated once each, lets say $C_3.1$ and $C_4.1$ respectively, else $step-10$ and $step-11$ will be evaluated once each, lets say $C_5.1$ and $C_6.1$ respectively. Otherwise $step-14$ will evaluated once, lets say $C_7.1$. At the end $step-7$ will be evaluated once, $C_8.1$. Hence we can write the recurance relation as given bellow.

$T(n) = T(\frac{n}{2}) + C_1.1 + C_2.1 + C_3.1 + C_4.1 + C_8.1$

$T(n) = T(\frac{n}{2}) + C$

For simplicity assume $n = 2^k$ and $C = 5$ in this case.

$T(n) = T(\frac{n}{2}) + 5$

$=T(\frac{n}{4}) + 5 + 5$

$=T(\frac{n}{8}) + 5 + 5 + 5$

.

.

.

$=T(\frac{n}{2^k}) + k.5$

$=T(1) + 5k$

as $k = \log_2 n$

$T(n) =C + 5\log_2 n$

$T(n) =O(log_2 n)$

If the binary search tree is formed only in one direction (left skewed or right skewed), then their may be the case where the best fit host will be in the leaf node, i.e. the last node. In that case the algorithm have to search all the n nodes of the host BST to find the best fit. So the worst case time complexity of the Allocate procedure is

$O(n)$.

Update Procedure:

As the deletion and insertion operation in a binary serach tree takes $O(n)$ in worst case and $O(log_2 n)$ in average case [29]. Hence in average case $step-3$ and $step-4$ will run for $O(log_2 n)$ number of times and in worst case it will run for $O(n)$ number of times. $step-3$ will run for $O(1)$ times in all the cases. Hence the average case time complexity for Update procedure is $O(log_2 n)$ and the worst case time complexity is $O(n)$.

VM Scheduler Algorithm:

Let m = number of virtual machines that have to be allocated to n physical host, Hence $step-2$ will run for m times in all cases. As finding the maximum element in a binary search tree takes $O(log_2 n)$ in average case and $O(n)$ in worst case [29]. Hence for each virtual machine $step-4$, $step-5$, $step-11$ and $step-12$ will run for $O(log_2 n)$ in average case and $O(n)$ in worst case. $step-3$ and $step-6$ will run for $O(1)$ times for each virtual machine in all cases. As we proved Allocate procedure will run for $O(log_2 n)$ in average case and $O(n)$ in worst case for each virtual machine. Hence for each virtual machine placement the VM Scheduler algorithm will run for $O(1) + O(log_2 n) + O(log_2 n) + O(1) + O(log_2 n) + O(log_2 n) + O(log_2 n)$ times i.e., $O(log_2 n)$ in average case and $O(1) + O(n) + O(n) + O(1) + O(n) + O(n) + O(n)$ times i.e., $O(n)$ in worst case. Hence for m virtual machines it will run for $O(m \times log_2 n)$ times in average case and $O(m \times n)$ times in worst case.

4.5 Conclusion

In this chapter the proposed scheme has been discussed. The proposed VM Sheduler algorithim is presented, which uses a best fit heuristic based on binary search tree. The time complexity analysis has also been discussed and it is found to be $O(m \times n)$ in worst case and $0(m \times \log_2 n)$ in average case.

Chapter 5

Simulation and Results

Experiment-1

Experiment-2

Experiment-3

Chapter 5

Simulation and Results

5.1 Introduction

The proposed VM Scheduler algorithm is simulated along with other three algorithms (Ranking algorithm, Greedy first fit algorithm and Round-Robin algorithm) using CloudSim [30].

CloudSIm is an open-source toolkit developed in Java, for simulation of cloud computing application and environment. CloudSim can be used to simulate data-center, Host machines, Virtual Machines, Cloudlet (a dummy application that will run on a virtual machine) etc. by using predefined classes and functions.

Using this simulator we have created some host machines in a data-center and some virtual machines to run on those host machines. We have implemented four VM provisioning models (VM Scheduler, Greedy First Fit, Ranking and Round-Robin) to allocate the created virtual machines to the host machines depending on the type of algorithm used in each.

The chapter 5 discuss the experiment datas and the results we obtained in section 5.2, 5.3 and 5.4.

5.2 Experiment-1:

Table 5.1: Experiment-1 Host memory specification

Host	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}
Memory (in MB)	2048	1024	2048	512	1024	1024	512	2048	1024	512

Table 5.2: Experiment-1 VM request memory specification

VM	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6	VM_7	VM_8	VM_9	VM_{10}
Memory (in MB)	256	2048	1024	512	1024	256	512	1024	512	1024

Results:

Table 5.3 shows the host machines in which the virtual machines of requested specification is created, by each placement policy.

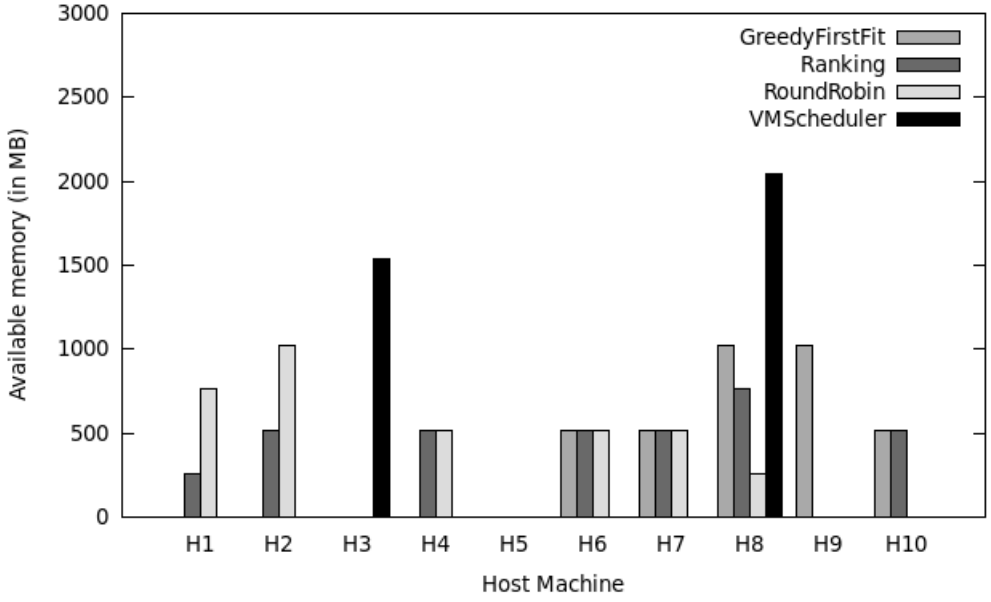

Figure 5.1: Experiment:1 Available Memory vs.Host Machine

Table 5.3: Experiment-1 VM to Host mapping

	Greedy First-Fit	Ranking	Round Robin	VM Scheduler
VM_1	H_1	H_1	H_1	H_3
VM_2	H_3	H_3	H_3	H_1
VM_3	H_1	H_8	H_5	H_2
VM_4	H_1	H_1	H_6	H_4
VM_5	H_2	H_1	H_8	H_5
VM_6	H_1	H_8	H_8	H_3
VM_7	H_4	H_2	H_8	H_7
VM_8	H_5	H_5	H_9	H_6
VM_9	H_6	H_6	H_{10}	H_{10}
VM_{10}	H_8	H_9	H_1	H_9

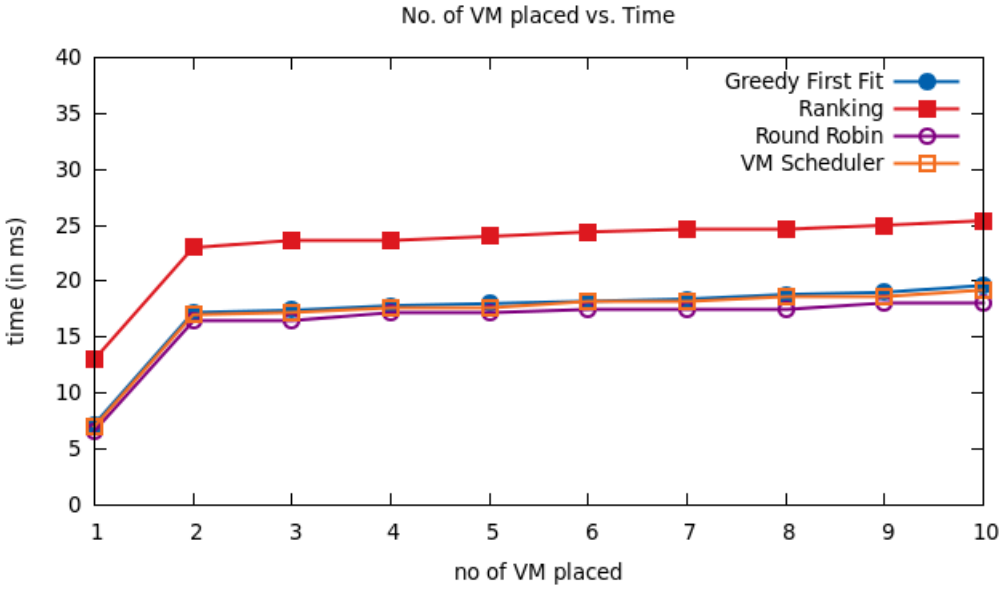

Figure 5.2: Experiment:1 Time vs. No. of VM placed

The graph in figure 5.1 shows that after allocation of all virtual machine requests, there are many small chunks of memory in different host machines in case of Greedy First Fit, Ranking and Round-Robin algorithm. But VM Scheduler utilizes all the host machines efficiently hence H3 and H8 is having two big chunks of memory which can be used to allocate two higher specification VM requests.

The graph in figure 5.2 shows that when number of VM request is not large in number, reedy First Fit, Round-Robin and VM scheduler have almost same time versus number of virtual machine placed graph, But Ranking algorithm is takes more time than the other three algorithm.

5.3 Experiment-2:

Table 5.4: Experiment-2 Host memory specification

Host	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}
Memory (in MB)	4096	2048	4096	1024	2048	4096	2048	2048	2048	1024

Table 5.5: Experiment-2 VM request memory specification

VM	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6	VM_7	VM_8	VM_9	VM_{10}
Memory (in MB)	2048	512	1024	1024	256	512	1024	512	256	2048
VM	VM_{11}	VM_{12}	VM_{13}	VM_{14}	VM_{15}	VM_{16}	VM_{17}	VM_{18}	VM_{19}	VM_{20}
Memory (in MB)	512	256	256	512	1024	512	512	2048	4096	4096

Results:

Form the graphs in figure 5.4 and 5.3, it can be conclude that VM scheduler managed to place all the VM requests by efficiently allocating these requests to the proper host machines. In case of other three placement policy the placement in not

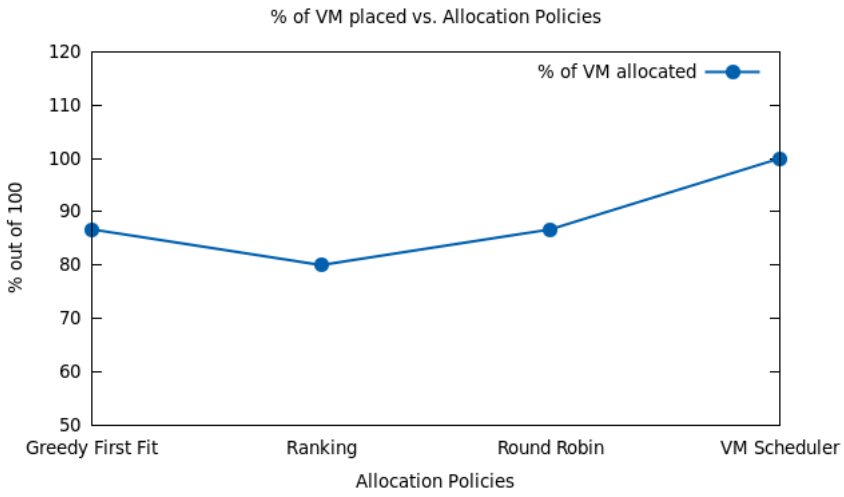

Figure 5.3: Experiment:2 % of VM allocated vs. Allocation Policies

100% even though some host having good chunks of memory, but not good enough to place the large VM requests. This shows the better performance of VM Scheduler compare to other three.

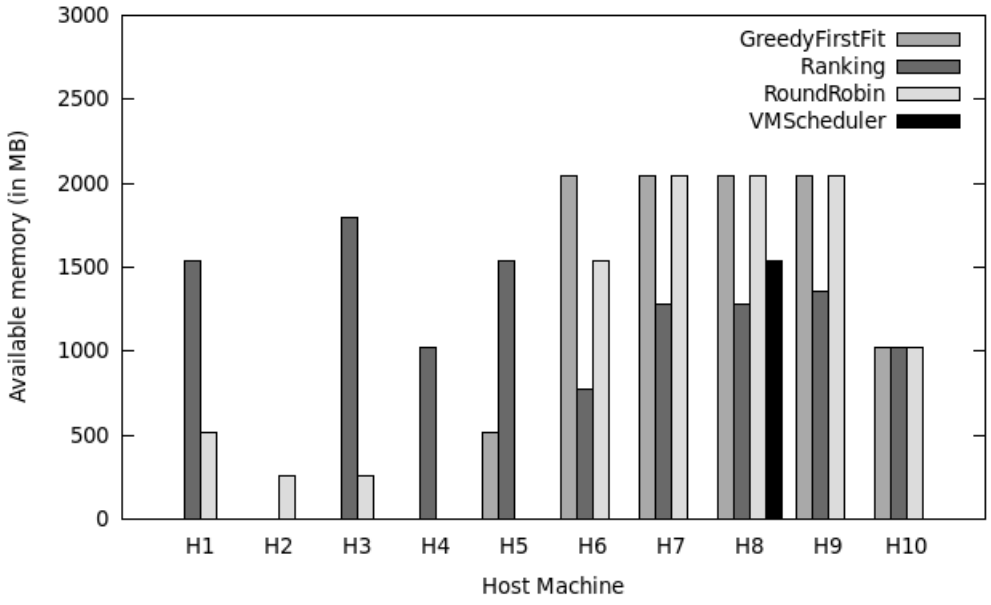

Figure 5.4: Experiment:2 Available Memory vs. Host Machine

5.4 Experiment-3:

Table 5.6: Experiment-3 Host memory specification

Host	H_{1-5}	H_{6-10}	H_{11-15}	H_{16-20}	H_{21-25}	H_{26-30}
Memory (in MB)	4096	2048	1024	4096	1024	2048

Table 5.7: Experiment-3 VM request memory specification

Host	VM_{1-10}	VM_{11-20}	VM_{21-30}	H_{31-35}	H_{36-40}	H_{41-50}	H_{51-55}
Memory (in MB)	2048	512	1024	512	1024	512	4096

The SLA violation percentage can be calculated as [100 - ($\frac{DeleveredService}{RequestedService} \times 100$) %] The graph in the figure 5.5 shows that even with increase in the number of

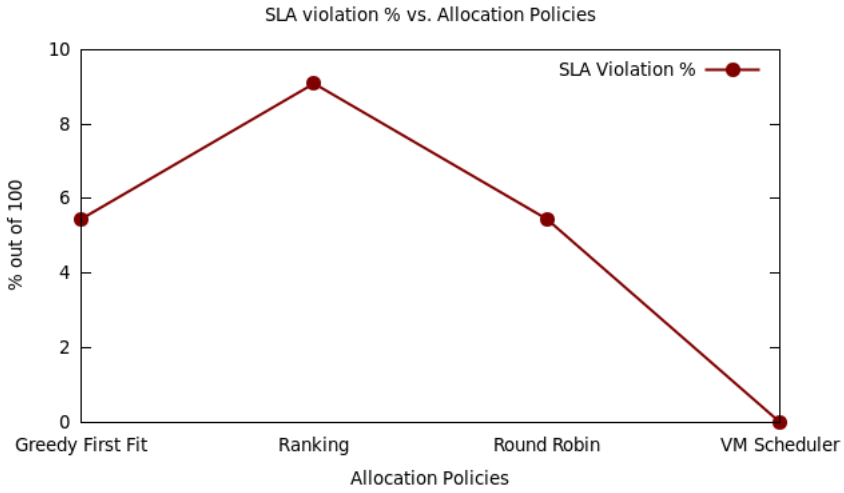

Figure 5.5: Experiment:3 SLA violation % vs. Allocation Policies

requests the SLA violation with respect to single dimensional request is zero in case of VM Scheduler. And graph in figure 5.6 shows that with increase in number of VM requests the running time of the VM Scheduler is less than that of other three.

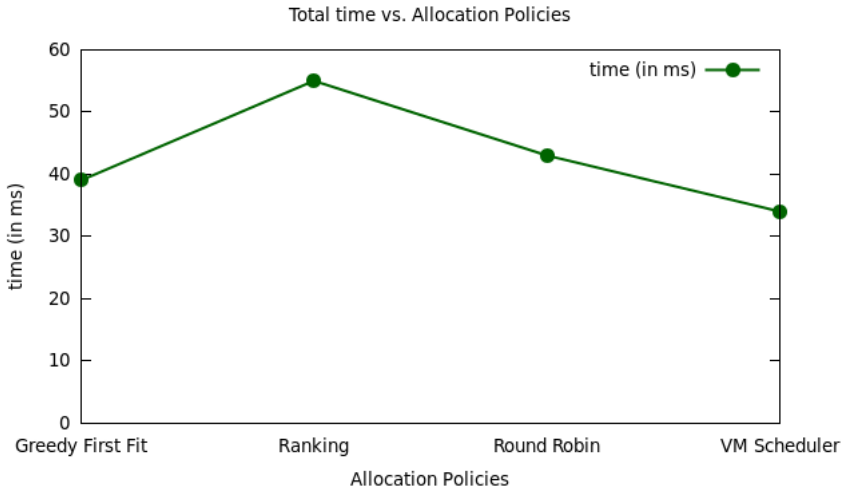

Figure 5.6: Experiment:3 Total time taken vs. Allocation Policies

5.5 Conclusion

The results obtained from the experiments carried out are in accordance with the proposed scheme and this shows that the proposed algorithm performs better than ranking, greedy first fit and round robin approches, taking minimizing cost, SLA violation into consideration.

Chapter 6

Conclusion

Conclusion

Scope for Further Research

Chapter 6

Conclusion

6.1 Conclusion

Virtual machine placement is an important issue in cloud computing, it is because all the requests that arrive for any infrastructure have to be served by creating virtual machines of the requested specification on the underlying physical machines. In case of On-Demand access the virtual machine requests have to served quickly for a small interval of time. In this paradigm to serve more requests at a particular time-frame, the physical machines should be used effectively i.e., the virtual machine placement policy shoud be good enough to minimize the number of physical machine used, considering the cost and SLA. In this thesis we discussed some virtual machine placement policies adopted by various open-source cloud computing solutions. We discussed our proposed framework for efficiently solve this problem, in which we described our proposed policy named VM Scheduler for virtual machine placement. From the results obtained it is clear that the proposed VM Scheduler is performing much better than other discussed placement policies in terms of minimizing cost, minimizing allocation time and minimizing SLA violations.

6.2 Scope for Further Research

In this work virtual machine migration has not been taken into consideration, by introducing which the performance can be improved further. After a virtual machine finishes its execution at a particular physical machine, that place can be taken by a more suitable virtual machine running on a different physical machine, which allows further minimization of number of physical resources used, and hence the cost.

Bibliography

[1] Shuai Zhang, Shufen Zhang, Xuebin Chen, and Xiuzhen Huo. Cloud computing research and development trend. In *Future Networks, 2010. ICFN'10. Second International Conference on*, pages 93–97. IEEE, 2010.

[2] Barrie Sosinsky. *Cloud computing bible*, volume 762. Wiley, 2012.

[3] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28, 2009.

[4] Piyush Patel and Arun Kr Singh. A survey on resource allocation algorithms in cloud computing environment. In *Golden Research Thoughts Volume 2, Issue. 4*. Ashok Yakkaldevi, 2012.

[5] Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.

[6] Bernard Golden. Cloud computing: Two kinds of agility. `www.cloudtweaks.com/2010/07/cloud-computing-two-kinds-of-agility/`, 2010.

[7] Multitenancy. `en.wikipedia.org/wiki/Multitenancy`, 2013.

[8] Cloud computing demystifying saas, paas and iaas. `www.cloudtweaks.com/2010/05/cloud-computing-demystifying-saas-paas-and-iaas/`, 2010.

[9] Ryan Sobel. Virtualization vs. cloud computing: There is a difference. `www.hightech-highway.com/virtualize/virtualization-vs-cloud-computing-there-is-a-difference/`, 2012.

[10] John Considine. Do vms still matter in the cloud? `www.cloudswitch.com/page/do-virtual-machines-still-matter-in-the-cloud`, 2010.

[11] Hyukho Kim, Woongsup Kim, and Yangwoo Kim. Experimental study to improve resource utilization and performance of cloud systems based on grid middleware. *Journal of Communication and Computer*, 7(12):32–43, 2010.

[12] K Mills, J Filliben, and C Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 91–98. IEEE, 2011.

[13] Bhanu P Tholeti. Hypervisors, virtualization, and the cloud: Learn about hypervisors, system virtualization, and how it works in a cloud environment. `www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/`, 2011.

[14] Understanding the virtualization market. `www.datacenterknowledge.com/archives/2012/08/01/hypervisor-101-a-look-hypervisor-market/`, 2012.

[15] Bhanu P Tholeti. Hypervisors, virtualization, and the cloud: Dive into the xen hypervisor. `www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare-xen/index.html`, 2011.

[16] Patrícia Takako Endo, Glauco Estácio Gonçalves, Judith Kelner, and Djamel Sadok. A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, 2010.

[17] Xen hypervisor. `http://www.xen.org/products/xenhyp.html`, 2012.

[18] About nimbus. `http://www.nimbusproject.org/about/`, 2012.

[19] Naidila Sadashiv and SM Dilip Kumar. Cluster, grid and cloud computing: A detailed comparison. In *Computer Science & Education (ICCSE), 2011 6th International Conference on*, pages 477–482. IEEE, 2011.

[20] About the opennebula.org project. `http://www.opennebula.org/about:about`, 2012.

[21] The eucalyptus cloud. `http://www.eucalyptus.com/eucalyptus-cloud/iaas`, 2013.

[22] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 119–128. IEEE, 2007.

[23] Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud. Autonomic virtual resource management for service hosting platforms. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8. IEEE Computer Society, 2009.

[24] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimal virtual machine placement across multiple cloud providers. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 103–110. IEEE, 2009.

[25] Hidemoto Nakada, Takahiro Hirofuchi, Hirotaka Ogawa, and Satoshi Itoh. Toward virtual machine packing optimization based on genetic algorithm. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 651–654. Springer, 2009.

[26] Shubham Agrawal, Sumit Kumar Bose, and Srikanth Sundarrajan. Grouping genetic algorithm for solving the serverconsolidation problem with conflicts. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 1–8. ACM, 2009.

[27] Borja Sotomayor, Rubén S Montero, Ignacio M Llorente, and Ian Foster. Virtual infrastructure management in private and hybrid clouds. *Internet Computing, IEEE*, 13(5):14–22, 2009.

[28] Hai Zhong, Kun Tao, and Xuejie Zhang. An approach to optimized resource scheduling algorithm for open-source cloud systems. In *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, pages 124–129. IEEE, 2010.

[29] Pawan Jindal, Amit Kumar, and Shishir Kumar. Analysis of time complexity in binary search tree. In *Proceedings of the 4th National Conference; INDIACom*, 2010.

[30] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

Dissemination

Journal

1. Sameer Kumar Mandal and Pabitra Mohan Khilar. Article: Efficient Virtual Machine Placement for On-Demand Access to Infrastructure Resources in Cloud Computing. *International Journal of Computer Applications* 68(12):6-11, April 2013. Published by Foundation of Computer Science, New York, USA.