

A NEW VIRTUAL MACHINE MIGRATION POLICY USING DYNAMIC THRESHOLD FOR CLOUDS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

INTEGRATED DUAL DEGREE

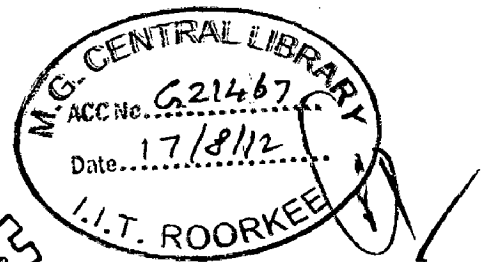
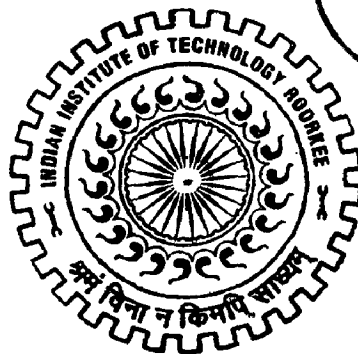
in

COMPUTER SCIENCE AND ENGINEERING

(With Specialization in Information Technology)

By

ANKIT AGARWAL



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

JUNE, 2012

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation work entitled "A New Virtual Machine Migration Policy Using Dynamic Threshold For Clouds" towards the partial fulfillment of the requirement for the award of the degree of **Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology)** and submitted to the **Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from May, 2011 to June, 2012 under the guidance and provision of **Dr. Manoj Misra, Professor, Department of Electronics and Computer Engineering, IIT Roorkee.**

I have not submitted the matter embodied in this dissertation work for the award of any other degree and diploma.

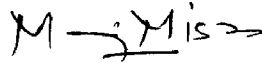
Date: June, 2012
Place: Roorkee


(Ankit Agarwal)

CERTIFICATE

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

Date: June, 2012
Place: Roorkee


Dr. Manoj Misra
Professor,
E&CE Department
IIT Roorkee, India

ACKNOWLEDGEMENTS

I would like to take this opportunity to extend my heartfelt gratitude to my guide and mentor **Dr. Manoj Misra, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee**, for his trust in my work, his able guidance, regular source of encouragement and assistance throughout this dissertation work. I would state that the dissertation work would not have been in the present shape without his inspirational support and I consider myself fortunate to have done my dissertation under him.

I also extend my sincere thanks to **Dr. Padam Kumar, Professor and Head of the Department of Electronics and Computer Engineering** for providing facilities for the work.

I would like to thank all my friends especially Rajdeep Barua who supported and encouraged me to finish this work.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing constant support, encouragement, and love.

ANKIT AGARWAL

ABSTRACT

Cloud computing is an emerging model of business computing. Cloud computing is the long dreamed vision of computing as a utility, where data owners can remotely store their data in the cloud to enjoy on-demand high-quality applications and services from a shared pool of configurable computing resources. As the cost of energy is increasing day by day, there is a need to shift to energy saving mechanism. In Cloud computing there are large number of host and different virtual machines running on them. In this dissertation, the concept of migration of virtual machines is studied. The different migration policies of the virtual machines and factors on which they depend are studied. The migration policy depends on two threshold values. The dynamic evaluation of these thresholds is suggested and a different migration policy is suggested. The performance evaluation is done on the basis of migration count of the virtual machines. Finally the thesis is concluded by pointing out future perspective of the migration policies and how can they be used in the real time environment.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	1
1.1. Cloud Computing.....	1
1.2. Problem Statement.....	2
1.2.1. Problem Description	2
1.3. Organization of Dissertation	3
CHAPTER 2: BACKGROUND AND LITERATURE REVIEW	4
2.1 Types of Cloud.....	4
2.1.1 Software as a Service (SaaS)	4
2.1.2 Platform as a Service (PaaS).....	5
2.1.3 Infrastructure as a Service (IaaS).....	6
2.2 Deployment Methods.....	7
2.2.1 Private Cloud	7
2.2.2 Public Cloud.....	7
2.2.3 Hybrid Cloud	7
2.3 CloudSim	8
2.3.1 Modelling in Cloud.....	10
2.3.2 Modelling in VM Allocation	11
2.4 Migration Policies.....	12
2.4.1 Minimization of Migration Policy	12
2.4.2 Highest Potential Growth Policy	14
2.4.3 Random Choice Policy	16

2.5	Dynamic Threshold.....	17
2.6	Roulette Wheel Selection.....	18
2.7	Research Gaps.....	20
CHAPTER 3: PROPOSED SCHEME.....		21
3.1	Proposed threshold policy.....	21
3.2	Proposed migration policy	23
CHAPTER 4: DESIGN AND IMPLEMENTATION		25
CHAPTER 5: RESULTS AND DISCUSSION		29
5.1	Evaluation of Threshold Policies using threshold policies.....	29
5.2	Evaluation of Migration Policies using proposed threshold policy	34
5.3	Evaluation of Proposed Threshold Policy.....	39
CHAPTER 6: CONCLUSION AND FUTURE WORK		41
6.1	Suggestions for future work.....	42
REFERENCES.....		43

LIST OF FIGURES

Figure 2.1	Layered CloudSim Architecture	9
Figure 2.2	Pseudo code for Minimization of Migration Policy	14
Figure 2.3	Pseudo code for Highest Potential Growth Policy	15
Figure 2.4	Pseudo code for Random Choice Policy	17
Figure 2.5	Roulette Wheel	19
Figure 3.1	Pseudo code for Modified Roulette Wheel Selection	24
Figure 4.1	CloudSim class diagram	25
Figure 5.1	The variation of virtual machine migration count with virtual machine count taking 50 host.....	30
Figure 5.2	The variation of virtual machine migration count with virtual machine count taking 100 host.....	31
Figure 5.3	The variation of virtual machine migration count with virtual machine count taking 150 host.....	31
Figure 5.4	The variation of virtual machine migration count with virtual machine count taking 200 host.....	32
Figure 5.5	The variation of virtual machine migration count with virtual machine count taking 250 host.....	33
Figure 5.6	The variation of virtual machine migration count with virtual machine count taking 300 host.....	34
Figure 5.7	The variation of virtual machine migration count with virtual machine count taking 50 host.....	35
Figure 5.8	The variation of virtual machine migration count with virtual machine count taking 100 host.....	36

CHAPTER 1

INTRODUCTION

1.1. Cloud Computing

Cloud computing is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on-demand. While it is sometimes considered simply an alternative means of traditional server or website hosting, the Cloud is actually much more than that, offering many different layers and opportunities. Cloud computing enables on demand network access to a shared pool of configurable computing resources. Of particular benefit is the flexible infrastructural platform that Cloud computing provides, which can change computing resources from a capital- and skill-intensive investment into an elastic-scale utility-model of allocation [1].

Cloud computing describes a data processing infrastructure in which the application software and often the data itself is stored permanently not on your PC but rather a remote server that's connected to the Internet. When you need to use the application or access the data, your computer connects to the server through the Internet and some of that information is cached temporarily on your client machine [1].

As energy costs are increasing there is a need to shift the focus from optimizing datacenter resource management for pure performance to optimizing them for energy efficiency, while maintaining high service level performance. Therefore, Cloud service providers need to adopt measures to ensure that their profit margin is not dramatically reduced due to high energy costs. There is also increasing pressure from governments worldwide aimed at the reduction of carbon footprints, which have a significant impact on the climate change. Thus, providers need to minimize energy consumption of Cloud infrastructures, while ensuring the service delivery [2].

1.2. Problem Statement

The aim is to evaluate performance of virtual machine migration policies using dynamic threshold allocation in cloud. and proposed a new policy.

1.2.1. Problem Description

In the real time environment there are large numbers of hosts and different virtual machines allocated to them. But many of these hosts can be shut down because only few virtual machines of lower CPU utilization are allocated to them.

The migration of virtual machines between nodes (hosts) is done, as most of the time the virtual machines don't use the resources allocated to them, thus they can be consolidated to a minimum number of nodes. The idle nodes can then be shut down or put to sleep, thus reducing the total energy consumption.

The migration policy takes two threshold values, ie lower utilization threshold and upper utilization threshold. These threshold values depend upon the current CPU utilization of the host. As the number of virtual machine allocated to the host changes, so does its utilization. The total CPU utilization of a particular host is the sum of CPU utilization of all the virtual machines allocated to it.

The goal stated in the problem statement can be divided into three sub-problems:

1. Dynamically determining the upper and lower utilization threshold values that will be used by the migration policy.
2. Performance evaluation of the existing migration policy by taking the dynamic threshold values (evaluated by both existing and suggested policy) and the static threshold values.
3. Performance evaluation of the proposed migration policy with the existing migration policy using dynamically determined threshold values.

1.3. Organization of Dissertation

This thesis report comprises of six chapters including this chapter that introduces the basic of cloud computing and states the problem. The rest of the dissertation report is organized as follows.

Chapter 2 provides a brief description of literature review on Cloud computing. It contains the literature review of CloudSim on which the simulation is performed. It also contains various migration policies along with the dynamic threshold evaluation policy.

Chapter 3 provides a detailed description of the proposed scheme for evaluating the dynamic threshold. It also contains the description of the proposed migration policy.

Chapter 4 gives the brief description of the implementation of the proposed scheme. It contains various classes in the CloudSim that is being used for the simulation.

Chapter 5 discusses the results and including discussion on them. The performance parameter used is the migration count.

Chapter 6 concludes the work and gives the directions for future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers. Cloud computing is a model for enabling 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. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models [3]

2.1. Types of Cloud

There are main three systems categories Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

2.1.1. Software as a Service (SaaS)

Traditionally, users prescribe software and it is license in and order to install it on their hard disk and then use it, however, in the cloud users do not required to purchase the software rather the payment will be based on pay-per-use model. It support multi-tenant which means that the physical backend infrastructure is shared among several users but logically it is unique for each user [4].

Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. There are two different delivery models for SaaS. First is the Hosted Application Management (Hosted AM) model is similar to ASP (Application Service Provider) in which a provider hosts commercially available software for customers and delivers it over the Web. Second is the Software on Demand model in which the provider

gives customers network-based access to a single copy of an application created specifically for SaaS distribution [4].

Benefits of the SaaS model include:

- Easier administration.
- Automatic updates and patch management.
- Compatibility: All users will have the same version of software.
- Easier collaboration.
- Global accessibility.

2.1.2. Platform as a Service (PaaS)

In PaaS the development environment are provided as service. The developers will use vendor's block of code to create their own applications. The platform will be hosted in the cloud and will be accessed using the browser [4].

It is a way to rent hardware, operating systems, storage and network capacity over the Internet. The service delivery model allows the customer to rent virtualized servers and associated services for running existing applications or developing and testing new ones. It is an outgrowth of Software as a Service (SaaS), a software distribution model in which hosted software applications are made available to customers over the Internet. PaaS has several advantages for developers. With PaaS, operating system features can be changed and upgraded frequently. Geographically distributed development teams can work together on software development projects. Services can be obtained from diverse sources that cross international boundaries. Initial and ongoing costs can be reduced by the use of infrastructure services from a single vendor rather than maintaining multiple hardware facilities that often perform duplicate functions or suffer from incompatibility problems. Overall expenses can also be minimized by unification of programming development efforts [4].

2.1.3. Infrastructure as Service (IaaS)

In IaaS [4], vendors provide the infrastructure as a service where it is delivered in form of technology, datacentres and IT services to the customer which is equivalent to the traditional “outsourcing” in the business world but with much less expenses and effort. The main purpose is to tailor a solution to the customer based on required applications.

Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it. The client typically pays on a per-use basis. Infrastructure as a Service is sometimes referred to as Hardware as a Service (HaaS) [4].

Table 2.1. : Cloud Computing Services

	Service	Providers
SaaS	Support running multiple instances of it. Develop software that is capable to run in the cloud.	Google Docs Mobile Me Zoho
PaaS	Platform which allows developer to create programs that can be run in the cloud. Includes several applications services which allow easy deployment.	Microsoft Azure Force.com Google App Engine.
IaaS	Highly scaled a shared and computing infrastructure accessible using internet browser. Consists of Database, servers and storage	Amazon S3 Sun’s Cloud Service

2.2. Deployment Methods

There are three deployment models for Cloud computing: Public, Private, and Hybrid.

2.2.1. Public Cloud

A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet. Public cloud services may be free or offered on a pay-per-usage model. The physical infrastructure is generally owned and managed by the service provider [1].

The main benefits of using a public cloud service are:

- Easy and inexpensive set-up because hardware, application and bandwidth costs are covered by the provider.
- Scalability to meet needs.
- No wasted resources because you pay for what you use.

Examples of public clouds include Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, Google AppEngine and Windows Azure Services Platform.

2.2.2. Private Cloud

A private cloud is a proprietary network or data center which is managed by the organization it serves. The physical infrastructure may be owned by and managed by the organization or the designated service provider with an extension of management and security control planes controlled by the organization [1].

2.2.3. Hybrid Cloud

A hybrid cloud is a composition of two or more Clouds (public or private) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability [1]. It is offered in one of two ways: a vendor has a private cloud and forms a partnership with a public cloud provider, or a public cloud provider forms a partnership with a vendor that provides private cloud platforms. It is a cloud computing environment in which an organization provides and manages some resources in-house and has others provided externally.

2.3. CloudSim

The CloudSim toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies.

CloudSim offers the following novel features [5]:

- (i) support for modeling and simulation of large-scale Cloud computing environments, including data centers, on a single physical computing node.
- (ii) a self-contained platform for modeling Clouds, service brokers, provisioning, and allocation policies.
- (iii) support for simulation of network connections among the simulated system elements.
- (iv) facility for simulation of federated Cloud environment that inter-networks resources from both private and public domains.
- (v) availability of a virtualization engine that aids in the creation and management of multiple, independent, and co-hosted virtualized services on a data center node.
- (vi) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

Figure 2.1. [5] shows the multi-layered design of the CloudSim software framework and its architectural components. The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues, such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state, are handled by this layer. A Cloud provider can study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), by implementing different strategies at this layer. Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to provisioning of hosts to VMs. A Cloud host can be concurrently allocated to a set of VMs that execute applications based on Software as a Service (SaaS) provider's defined Quality of Service (QoS) levels. This layer also exposes the functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study.

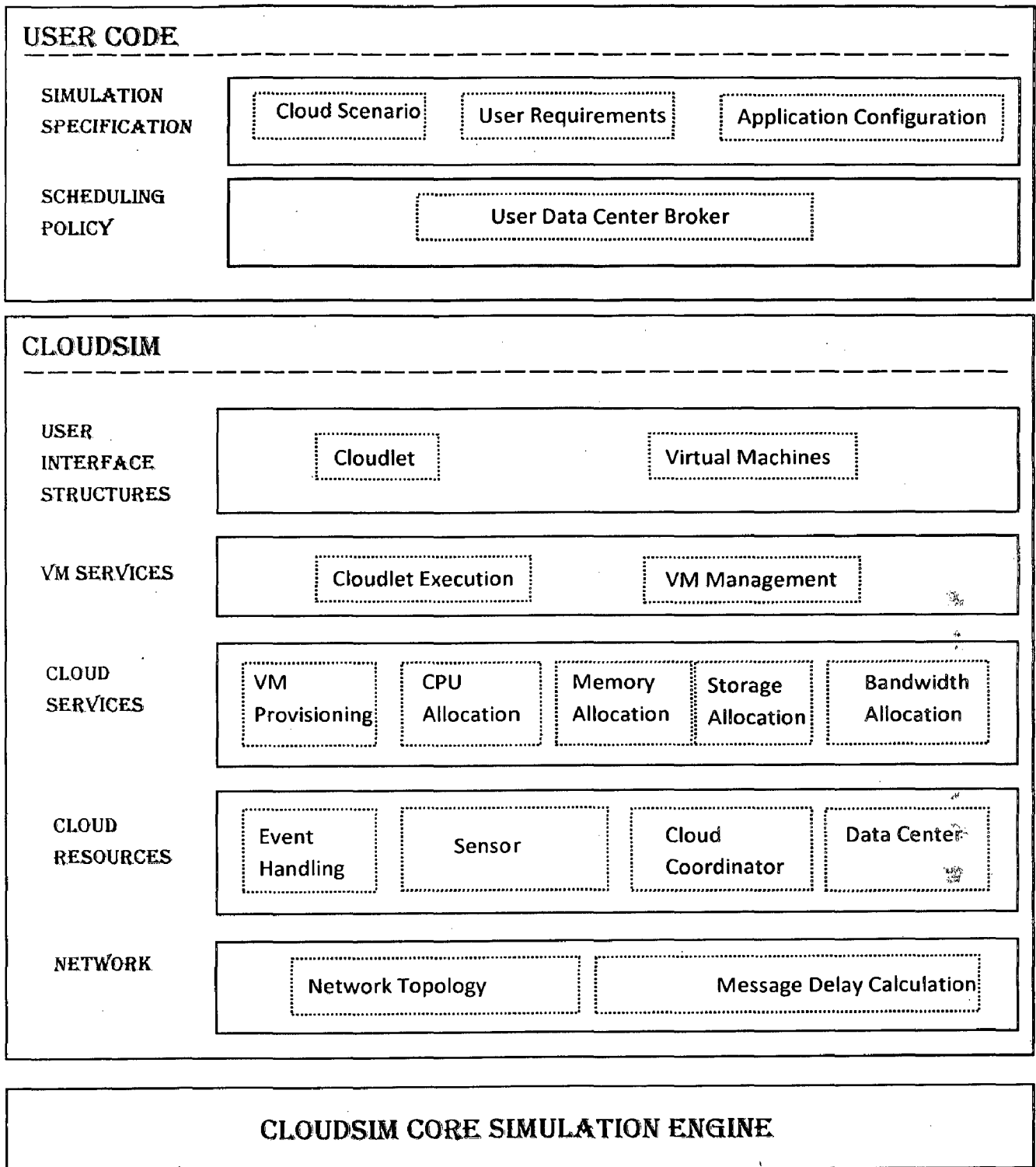


Figure 2.1. : Layered CloudSimArchitecture[5]

The top-most layer [5] in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities given at this layer, a Cloud application developer can perform the following activities: (i) generate a mix of workload request distributions, application configurations; (ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; and (iii) implement custom application provisioning techniques for clouds and their federation.

2.3.1. Modelling in Cloud

The infrastructure-level services (IaaS) [5] related to the clouds can be simulated by extending the data center entity of CloudSim. The data center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider. The VM policy stands for the operations control policies related to VM life cycle such as: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. Similarly, one or more application services can be provisioned within a single VM instance, referred to as application provisioning in the context of Cloud computing. In the context of CloudSim, an entity is an instance of a component. A CloudSim component can be a class (abstract or complete) or set of classes that represent one CloudSim model (data center, host).

A data center [5] can manage several hosts that in turn manage VMs during their life cycles. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

VM allocation (provisioning) [5] is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider. CloudSim supports the development of

custom application service models that can be deployed within a VM instance and its users are required to extend the core Cloudlet object for implementing their application services. Furthermore, CloudSim does not enforce any limitation on the service models or provisioning techniques that developers want to implement and perform tests with. Once an application service is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy. Allocation of application-specific VMs to hosts in a Cloud-based data center is the responsibility of a VM Allocation controller component (called *VmAllocationPolicy*). By default, *VmAllocationPolicy* implements a straightforward policy that allocates VMs to the Host on a First-Come-First-Serve (FCFS) basis. Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning. Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in CloudSim.

For each Host component,[5] the allocation of processing cores to VMs is done based on a hostallocation policy. This policy takes into account several hardware characteristics, such as numberof CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated toa given VM instance. Hence, CloudSim supports simulation scenarios that assign specific CPUcores to specific VMs (a space-shared policy), dynamically distribute the capacity of a core amongVMs (time-shared policy), or assign cores to VMs on demand.

2.3.2. Modelling the Vm Allocation

Clouds [5] contain an extra layer (the virtualization layer) that acts as an execution, management, and hosting environment for application services. Hence, traditional application provisioning models that assign individual application elements to computing nodes do not accurately representthe computational abstraction, which is commonly associated with Cloud resources. For example,consider a Cloud host that has a single processing core. There is a requirement of concurrentlyinstantiating two VMs on that host. Although in practice VMs are contextually (physical andsecondary memory space) isolated, still they need to share the processing cores and system bus.Hence, the amount of hardware resources available to each VMis constrained by the total processingpower and system bandwidth available within the host. This critical factor must be consideredduring the VM provisioning process, to avoid

creation of a VM that demands more processing power than is available within the host. In order to allow simulation of different provisioning policies under varying levels of performance isolation, CloudSim supports VM provisioning at two levels: first, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM. At the VM level, the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine.

2.4. Migration Policy

There are large numbers of Hosts and different Virtual Machines are allocated to them. In the simulation process using CloudSim, the virtual machines might not get the complete CPU utilization what it is expecting. The host CPU utilization might be less than the maximum which can be allocated to it. The measure at any instance of time of the CPU utilization of host is the sum of CPU utilization of all the virtual machines allocated to that host. Many of the hosts can be shut down because only few virtual machines of lower CPU utilization are allocated to them. These virtual machines can be migrated to other hosts.

To determine when and which VMs should be migrated, the algorithm uses a double-threshold VM selection policies. The basic idea is to set upper and lower utilization thresholds for hosts and keep the total utilization of the CPU by all the VMs allocated to the host between these thresholds. If the CPU utilization of a host falls below the lower threshold, all VMs have to be migrated from this host and the host has to be switched to the sleep mode in order to eliminate the idle power consumption. If the utilization exceeds the upper threshold, some VMs have to be migrated from the host to reduce the utilization [2].

2.4.1. The Minimization of Migration Policy

The Minimization of Migrations (MM) policy selects the minimum number of VMs needed to migrate from a host to lower the CPU utilization below the upper utilization threshold if the upper threshold is violated [2].

Let V_j be a set of VMs currently allocated to the host j . Then $P(V_j)$ is the power set of V_j . The MM policy finds a set $R \in P(V_j)$ as shown in equation (2.1) [2] where T_u is the upper utilization threshold; T_l is the lower utilization threshold; u_j is the current CPU utilization of the host j ; and $u_a(v)$ is the fraction of the CPU utilization allocated to the VM v .

$$R = \begin{cases} \{S \mid S \in P(V_j), u_j - \sum_{v \in S} u_a(v) < T_u, |S| \rightarrow \min\} & \text{if } u_j > T_u; \\ V_j & \text{if } u_j < T_u; \\ \emptyset & \text{Otherwise} \end{cases} \quad (2.1.)$$

As shown in the figure 2.2, the algorithm sorts the list of VMs in the decreasing order of the CPU utilization. Then, it repeatedly looks through the list of VMs and finds a VM that is the best to migrate from the host. The best VM is the one that satisfies two conditions. First, the VM should have the utilization higher than the difference between the host's overall utilization and the upper utilization threshold. Second, if the VM is migrated from the host, the difference between the upper threshold and the new utilization is the minimum across the values provided by all the VMs.

If there is no such a VM, the algorithm selects the VM with the highest utilization, removes it from the list of VMs, and proceeds to a new iteration. The algorithm stops when the new utilization of the host is below the upper utilization threshold. The complexity of the algorithm is proportional to the product of the number of over-utilized hosts and the number of VMs allocated to these hosts [2].

```

Input: hostList
Output: migrationList

for ( each Host h in hostList ) {
    vmList = h.getVmList();
    vmList.sortDecreasingUtilization();
    hUtil = h.getUtil();
    bestFitUtil = MAX;
    while (hUtil > THRES_UP){
        for( each vm v in vmList){
            if(v.getUtil() > hUtil - THRES_UP){
                t = v.getUtil() - hUtil + THRES_UP;
                if ( t < bestFitUtil ) {
                    bestFitUtil = t;
                    bestFitVm = v;
                }
            }
            else if ( bestFitUtil = MAX)
                bestFitVm = v;
            break;
        }
        hUtil = hUtil - bestFitUtil;
        migrationList.add(bestFitVm);
        vmList.remove(bestFitVm);
    }
    if(hUtil < THRES_LOW){
        migrationList.add(h.getVmList());
        vmList.remove (vmList);
    }
}
return migrationList;

```

Figure 2.2. : Pseudo code for Minimization of Migration Policy[2]

2.4.2. The Highest Potential Growth Policy

When the upper threshold is violated, the Highest Potential Growth (HPG) policy migrate VMs that have the lowest usage of the CPU relatively to the CPU capacity defined by the VM parameters in order to minimize the potential increase of the host's utilization [2].

As shown in equation (2.2) [2], $u_r(v)$ is the fraction of the CPU capacity initially requested for the VM v and defined as the VM's parameter.

$$R = \begin{cases} \{S \mid S \in P(V_j), u_j - \sum_{v \in S} u_a(v) < T_u, \sum \frac{u_a(v)}{u_r(v)} \Rightarrow \min\} & \text{if } u_j > T_u; \\ V_j & \text{if } u_j < T_u; \\ \emptyset & \text{Otherwise} \end{cases} \quad (2.2)$$

```

Input :hostList
Output: migrationList

for ( each host h in hostList){
    vmList = h.getVMList()
    hostUtil = h.getUtil ();

    if ( hostUtil<= Thres_Low) {
        migrationList.add(vmList);
    }
    else if ( hostUtil>Thres_Up){
        vmList.sortIncreasingRelativeUtil();
        for ( each Vm v in vmList){
            if (hostUtil>Thres_Up){
                vmUtil = v.getUtil();
                if ( vmUtil> (hostUtil - upperThreshold)){
                    migrationList.add(v);
                    hostUtil -= vmUtil;
                    vmList.remove(v);
                }
            }
        }
    }
}
return migrationList;

```

Figure 2.3. Pseudo Code for Highest Potential growth Policy

As shown in the figure 2.3., the algorithm finds the host utilization and if it is lesser than the lower utilization threshold then all the virtual machines allocated to that host is added to the migration list. If the host utilization is more than the upper threshold then the virtual machines are sorted in the increasing order of their relative utilization (the ratio between the allocated utilization to the actual utilization). The virtual machine having the lower relative utilization is added to the migration list and is removed from the virtual machine list. The process continues till the host utilization is more than the upper threshold value.

2.4.3. The Random Choice Policy

The Random Choice (RC) policy relies on a random selection of a number of VMs needed to decrease the CPU utilization by a host below the upper utilization threshold.

According to a uniformly distributed discrete random variable (X), whose values index subsets of V_j , the policy selects a set $R \in P(V_j)$, as shown in equation (2.3) [3].

$$R = \begin{cases} \{S \mid S \in P(V_j), u_j - \sum_{v \in S} u_a(v) < T_u, X \cong U(0, |P(V_j)| - 1)\} & \text{if } u_j > T_u; \\ V_j & \text{if } u_j < T_u \quad (2.3.) \\ \emptyset & \text{Otherwise} \end{cases}$$

As shown in the Figure 2.3., the algorithm finds the host utilization and if it is lesser than the lower utilization threshold then all the virtual machines allocated to that host is added to the migration list. If the host utilization is more than the upper threshold then the randomly a virtual machine is selected and is added to the migration list. This virtual machine is then removed from the virtual machine list. The process continues till the host utilization is more than the upper threshold value.

```

Input: hostUtil
Output: migrationList

for (each host in hostList){
    vmList = h.getVMList()
    hostUtil = h.getUtil ();

    if ( hostUtil<= Thres_Low) {
        migrationList.add(vmList);
    }
    else if ( hostUtil>Thres_Up){
        for ( each Vm v in vmList ){
            if ( hostUtil>Thres_Up) {
                Vmvm= vmList.get(Random(vm.size()));
                vmUtil = vm.getUtil();
                if ( vmUtil> (hostUtil - upperThreshold)){
                    migrationList.add(vm);
                    hostUtil -= vmUtil;
                    vmList.remove(vm);
                }
            }
        }
    }
}
return migrationList;

```

Figure 2.4. : Pseudo Code for Random Choice Policy

2.5. Dynamic Threshold

The dynamic threshold (DT) is used because the fixed values for the thresholds are unsuitable for an environment with dynamic and unpredictable workloads, in which different types of applications can share a physical resource. The system should be able to automatically adjust its behavior depending on the workload patterns exhibited by the applications. Therefore automated adjustment of the utilization thresholds based on a statistical analysis of the historical data collected during the lifetime of VMs is used [2].

The calculation of CPU Utilization of Virtual Machine is done [6] using equation (2.4).

$$U_{vm} = \frac{\text{totalRequestedMips}}{\text{totalMips for that VM}} \quad (2.4)$$

The Upper Utilization Threshold is done as follows:

$$\text{sum} = \sum U_{vm} \quad (2.5.)$$

$$\text{sqr} = \sqrt{\sum U_{vm}} \quad (2.6.)$$

$$T_{upper} = 1 - (((P_{uu} * \text{sqr}) + \text{sum}) - ((P_{ul} * \text{Sqr}) + \text{sum})) \quad (2.7.)$$

where, for each host we preserve amount of CPU capacity by upper (P_{uu}) and lower (P_{ul}) probability limits.

The Lower Utilization Threshold is done as follows:

$$\text{sum} = \frac{\sum U_{vm}}{n} \quad (2.8.)$$

$$\text{sqr} = \frac{\sqrt{\sum (U_{vm} - \text{sum})^2}}{n} \quad (2.9.)$$

$$T_{lower} = \begin{cases} \text{sum} - (P_l * \text{sqr}) & \text{if CPU utilization is } < 30\% \\ 0.3 & \text{if CPU utilization is } \geq 30\% \end{cases} \quad (2.10.)$$

where, we considered P_l as probability limit of lower threshold and n is number VMs on the host.

2.6. Roulette Wheel Selection

Roulette Wheel Selection (RWS) [7] scheme probabilistically select individual based on their fitness value F_i . The circumference of Roulette-wheel is the sum of fitnesses of all the individuals. The individual whose fitness is larger is more probable to get selected. As shown in the figure 2.5. , the probability of area A getting selected is higher than the all other areas.

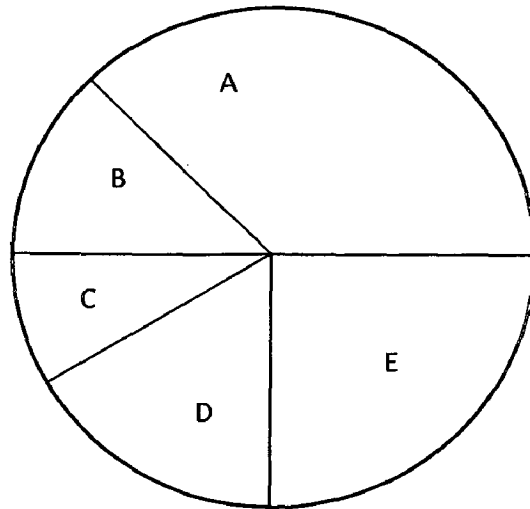


Figure 2.5. Roulette Wheel

The probability of Area_i to get selected is

$$p_i = \frac{F_i}{\sum F_i} \quad (2.11.)$$

The cumulative probability S is given as

$$S = \sum p_i \quad (2.12.)$$

The cumulative probability of an individual lies between 0 and 1. The i^{th} individual represents the cumulative probability from p_{i-1} to p_i . Thus in order to select n individuals, n random numbers between 0 and 1 are generated. The individual that represents the chosen random number in the cumulative probability range gets selected. This way, the individual with higher fitness value will represent a larger range in the cumulative probability values and therefore has a higher probability of getting selected. On the other hand, an individual with smaller fitness value represents a smaller range in cumulative probability values and has a smaller probability of getting selected.

2.7. Research Gaps

The following research gaps were identified after the literature review:

1. The policy for determining the upper and lower utilization threshold takes probability limit into account which is not valid for the real time applications. A policy for determining the threshold values is suggested taken into consideration both the virtual machine that are allocated and not allocated.
2. The policy given for the determining the migration count of virtual machines takes into account the virtual machines whose utilization is largest. A policy is suggested to determine which virtual machine needs to be migrated taking into account other virtual machines allocated to the host.

CHAPTER 3

PROPOSED SCHEME

This chapter gives the proposed schemes for dynamic threshold evaluation and for the migration count of virtual machines.

3.1. Proposed Threshold Policy (PT)

The lower and upper utilization threshold used in the migration policy should be dynamically determined. In real time environment there are large number of hosts and virtual machines. There are different virtual machines allocated to the host each having different CPU utilization which might be different than the actual CPU utilization of virtual machines. Thus having a same static threshold value for every host is not efficient. Thus at each host depending upon its utilization, its threshold value need to be determined. These threshold values depend upon the virtual machines allocated to the host. The total utilization of the host is sum of all the virtual machine's utilization allocated to it.

Let there be total N virtual machines which can be allocated to H hosts. The actual (requested) CPU utilization of the N virtual machines is X_1, X_2, \dots, X_N . At any instance of time t let there be n virtual machines allocated to H_i host. The CPU utilization of the host H_i is the sum of utilization of these n virtual machines ie the current CPU utilization of the virtual machines, not its actual CPU utilization.

Define a Utilization Vector UV of size N as follows:

for each i from 1 to N

$$\begin{aligned} UV_i &= 0 \text{ if } VM_i \text{ is not allocated at time } t \\ &= x_i \text{ if } VM_i \text{ is allocated at time } t \text{ where } x_i \text{ is its CPU utilization at time } t \end{aligned} \tag{3.1.}$$

Define Error Vector EV of size N as follows:

for each i from 1 to N

$$EV_i = \frac{(x_i(t) - x_i(t-1))}{X_i} \quad (3.2.)$$

where,

$x_i(t)$ is the CPU utilization of the allocated VM_i at time t

$x_i(t-1)$ is the CPU utilization of the allocated VM_i at time t-1

($x_i(t)$ and $x_i(t-1)$ can be zero if at time t or t-1 VM_i is not allocated)

X_i is the actual (requested) CPU utilization of the allocated VM_i

The net error = Utilization Vector * Error Vector

Example:

Let there be 5 virtual machine VM₁, VM₂, VM₃, VM₄, VM₅ with actual CPU utilization as X_1, X_2, X_3, X_4 and X_5 .

At time t VM₁, VM₂ and VM₃ are allocated with CPU Utilization as $x_1(t), x_2(t), x_3(t)$.

At time t+1 VM₂ and VM₄ are allocated with CPU Utilization as $x_2(t+1), x_4(t+1)$.

$$\text{Utilization Vector (UV)} = [0 \quad x_2(t+1) \quad 0 \quad x_4(t+1) \quad 0] \quad (3.3.)$$

$$\text{Error Vector (EV)} = \left[\frac{(0 - x_1(t))}{X_1} \quad \frac{(x_2(t+1) - x_2(t))}{X_2} \quad \frac{(0 - x_3(t))}{X_3} \quad \frac{(x_4(t+1) - 0)}{X_4} \quad \frac{(0 - 0)}{X_5} \right] \quad (3.4.)$$

Net Error = Utilization Vector * Error Vector

$$= 0 * \frac{(0 - x_1(t))}{X_1} + x_2(t+1) * \frac{(x_2(t+1) - x_2(t))}{X_2} + 0 * \frac{(0 - x_3(t))}{X_3} + x_4(t+1) * \frac{(x_4(t+1) - 0)}{X_4} + 0 * \frac{(0 - 0)}{X_5} \quad (3.5.)$$

At the start of the simulation, let there be n Virtual Machines be allocated to Host H_i .

$$\text{Sum} = \sum \text{Utilization of } n \text{ Virtual Machines allocated to } H_i \quad (3.6.)$$

$$\text{Lower Threshold Value} = a * \text{Sum} \quad (3.7.)$$

$$\text{Upper Threshold Value} = b * \text{Sum} \quad (3.8.)$$

$$a + b = 1 \quad (3.9.)$$

where, a and b are positive constants

Thus,

$$\text{Lower Threshold Value (t)} = \text{Total Error} + \text{Lower Threshold Value (t-1)} \quad (3.10.)$$

$$\text{Upper Threshold Value (t)} = \text{Total Error} + \text{Upper Threshold Value (t-1)} \quad (3.11.)$$

So initially the lower and upper utilization threshold are taken as some fraction of the total CPU utilization of host and at every simulation time it gets modified taking into consideration the error in the utilization. Thus each host has lower and upper utilization threshold value according to the virtual machines allocated to that host. At every simulation time these values are modified.

3.2. Proposed Migration Policy

The proposed algorithm is the modification of Roulette Wheel Selection (MRWS)

There are N Virtual Machines and H Hosts. Suppose at any given instance of time t n virtual machines are allocated to the host H_i and the upper and lower utilization threshold value be L and U .

The CPU utilization of the these n virtual machines be x_1, x_2, \dots, x_n where $x_1 < x_2 < \dots < x_n$.

As shown in the Figure 3.1., the algorithm first find the cumulative of the relative utilization of each of the virtual machine allocated to the host. A random number is selected between 0

and 1. If this number is greater than cumulative of the relative utilization, then this virtual machine is selected. This virtual machine is added to the migration list and is removed from the virtual machine list. The process continues till the host utilization is greater than the upper threshold value. If the host utilization is greater than the lower threshold then all the virtual machines allocated to the host is added to the migration list.

```

Input: hostList
Output: migrationList

for ( each Host h in hostList ) {
    vmList = h.getVmList();
    vmList.sortIncreasingUtilization();

    for( each vmv in vmList)
        sum += v.getUtil()

    for( each vmv in vmList){
        pi = v.getUtil()/sum
        ci = ci-1 + pi;
    }
    while (sum >Thres_Up){
        number = Random number between 0 and 1
        for( each vmv in vmList){
            if(number > ci) {
                migrationList.add (v)
                sum = sum - v.getUtil()
                vmList.remove (v)
                break
            }
        }
        if (sum <Thres_Low){
            for( each vm v in vmList){
                migrationList.add(v)
                sum = sum - v.getUtil()
                vmList.remove(v)
            }
        }
    }
    if (sum <Thres_Low)
        migartionList.add(vmList)
}
return migrationList

```

Figure 3.1. Pseudo Code for Modified Roulette Wheel Selection

CHAPTER 4

DESIGN & IMPLEMENTATION

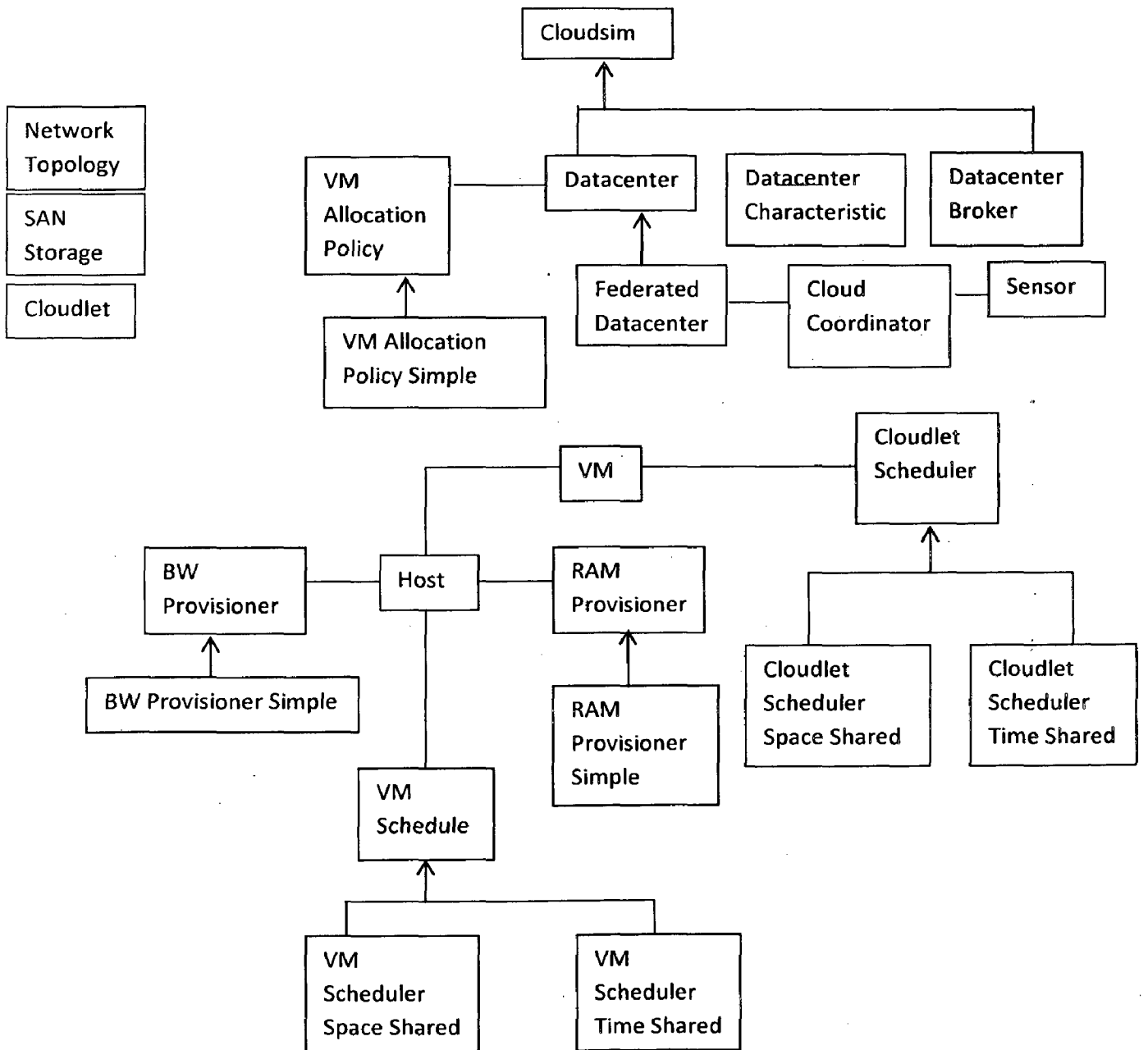


Figure 4.1. : CloudSim class design diagram [5]

621467
 12/8/12

The class design diagram of CloudSim [5] is shown in the figure 4.1. There are various classes that are used in the simulation process.

BwProvisioner: This is an abstract class that models the policy for provisioning of bandwidth to VMs. The main role of this component is to undertake the allocation of network bandwidth to a set of competing VMs that are deployed across the data center. Cloud system developers and researchers can extend this class with their own policies (priority, QoS) to reflect the needs of their applications. The BwProvisioningSimple allows a VM to reserve as much bandwidth as required; however, this is constrained by the total available bandwidth of the host [5].

CloudCoordinator: This abstract class extends a Cloud-based data center to the federation. It is responsible for periodically monitoring the internal state of data center resources and based on that it undertakes dynamic load-shedding decisions. Concrete implementation of this component includes the specific sensors and the policy that should be followed during load-shedding. Monitoring of data center resources is performed by the updateDatacenter() method by sending queries Sensors. Service/Resource Discovery is realized in the setDatacenter() abstract method that can be extended for implementing custom protocols and mechanisms (multicast, broadcast, peer-to-peer) [5].

Cloudlet: This class models the Cloud-based application services (such as content delivery, social networking, and business workflow). CloudSim orchestrates the complexity of an application in terms of its computational requirements. Every application service has a pre-assigned instruction length and data transfer (both pre and post fetches) overhead that it needs to undertake during its life cycle. This class can also be extended to support modeling of other performance and composition metrics for applications such as transactions in database-oriented applications [5].

CloudletScheduler: This abstract class is extended by the implementation of different policies that determine the share of processing power among Cloudlets in a VM. There are two types of provisioning policies are offered: space-shared (CloudletSchedulerSpaceShared) and time-shared (CloudletSchedulerTimeShared) [5].

Datacenter: This class models the core infrastructure-level services (hardware) that are offered by Cloud providers. It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations (memory, cores, capacity, and storage). Furthermore, every Datacenter component instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices to hosts and VMs [5].

DatacenterBroker or Cloud Broker: This class models a broker, which is responsible for mediating negotiations between SaaS and Cloud providers; and such negotiations are driven by QoS requirements. The broker acts on behalf of SaaS providers. The difference between the broker and the CloudCoordinator is that the former represents the customer (i.e. decisions of these components are made in order to increase user-related performance metrics), whereas the latter acts on behalf of the data center, i.e. it tries to maximize the overall performance of the datacenter, without considering the needs of specific customers [5].

DatacenterCharacteristics: This class contains configuration information of data center resources [5].

Host: This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (to represent a multi-core machine), an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs [5].

NetworkTopology: This class contains the information for inducing network behavior (latencies) in the simulation [5].

RamProvisioner: This is an abstract class that represents the provisioning policy for allocating primary memory (RAM) to the VMs. The execution and deployment of VM on a host is feasible only if the RamProvisioner component approves that the host has the required amount of free memory. The RamProvisionerSimple does not enforce any limitation on the amount of memory that a VM may request. However, if the request is beyond the available memory capacity, then it is simply rejected [5].

SanStorage: This class models a storage area network that is commonly ambient in Cloud-based data centers for storing large chunks of data. SanStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, subject to the availability of network bandwidth. Accessing files in a SAN at run-time incurs additional delays for task unit execution; this is due to the additional latencies that are incurred in transferring the data files through the data center internal network [5].

Sensor: This interface must be implemented to instantiate a sensor component that can be used by a CloudCoordinator for monitoring specific performance parameters (energy-consumption, resource utilization). CloudCoordinator utilizes the dynamic performance information for undertaking load-balancing decisions. The methods defined by this interface are: (i) set the minimum and maximum thresholds for performance parameter and (ii) periodically update the measurement [5].

Vm: This class models a VM, which is managed and hosted by a Cloud host component. Every VM component has access to a component that stores the following characteristics related to a VM: accessible memory, processor, storage size, and the VM's internal provisioning policy that is extended from an abstract component called the CloudletScheduler [5].

VmmAllocationPolicy: This abstract class represents a provisioning policy that a VM Monitor utilizes for allocating VMs to hosts. The chief functionality of the VmmAllocationPolicy is to select the available host in a data center that meets the memory, storage, and availability requirement for a VM deployment [5].

VmScheduler: This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processor cores to VMs. The functionalities of this class can easily be overridden to accommodate application-specific processor sharing policies [5].

CHAPTER 5

RESULTS

The simulation is done using CloudSim.

Each host is modeled to have one CPU core, with 8 GB RAM and 1 TB storage and 1000, 2000, 3000 MIPS which is randomly allocated. The virtual machines are been assigned CPU utilization as 250, 500, 750, 1000 MIPS randomly. Virtual machines were created each requires one CPU core with 128 RAM and 2 GB storage. Each virtual machine runs an application with 150,000 MI ie 10 minutes of the execution on 250 MIPS CPU with 100% utilization. In case of static threshold the lower utilization threshold is taken as 1000 MIPS (lowest CPU utilization of host) and higher utilization threshold to be 1500 MIPS (average CPU utilization of host).

The simulation is done with varying the number of host from 50 to 300 hosts.

5.1. Evaluation of the Migration Policies using Threshold Policies

In this section we present the result of the evaluation of three different threshold policies used in the different migration policies. The value of lower and upper utilization threshold for the static threshold policies is taken as 1000MIPS and 1500MIPS respectively. The value of constants a and b used in the proposed threshold policy is taken as 0.33 and 0.67 respectively.

The three different threshold policies are:

1. Static Threshold Values
2. Dynamic threshold policy
3. Proposed threshold policy

The figure 5.1 shows the variation of migration count with the virtual machines. The simulation is done taking 50 hosts. The MM policy using proposed threshold schemes gives the lowest migration count. The MM policy using the given dynamic threshold gives the highest migration count. The overall migration count increases as the virtual machine count

increases. There are various dips seen in the graph. It is due to the fact that as the number of host is taken as 50, and different number of virtual machines are allocated to them, most of which are not allocated as the specifications are not met. The lines in the graph are also close by because the difference in the migration count is less (around 100-250).

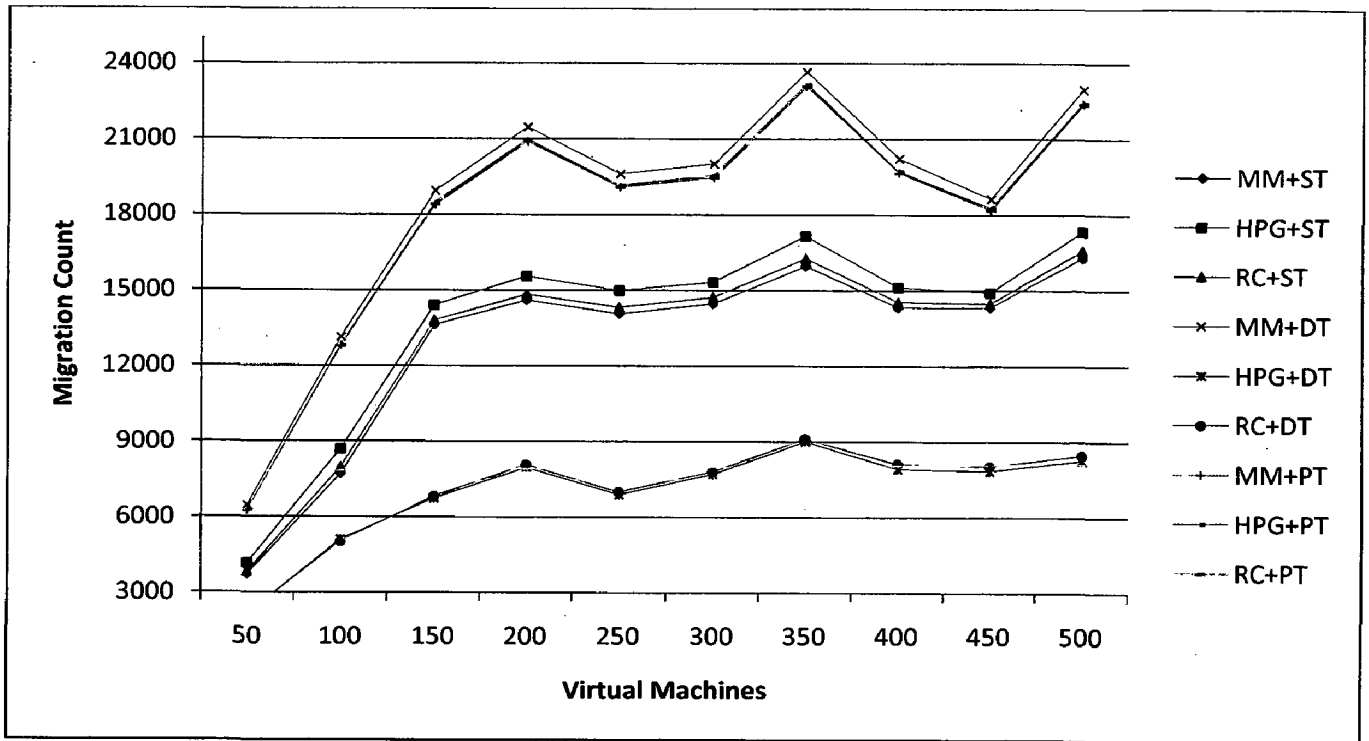


Figure 5.1. : The variation of virtual machine migration count with virtual machine count taking 50 hosts

The figure 5.2 shows the variation of migration count with the virtual machines. The simulation is done taking 100 hosts. The RC policy using dynamic threshold schemes gives almost same result as RC policy using static threshold. The MM policy using the given DT policy gives the highest migration count. The overall migration count increases as the virtual machine count increases. The dip in the MM using DT policy around 450 virtual machine is because in the PT policy the values of a and b constant are used for evaluation of the threshold values which in turn depends upon the current utilization of the host and previous threshold values, which is not in the case of DT and ST policy. Thus the graph shows almost straight lines in case of the migration policy using DT threshold evaluation policy.

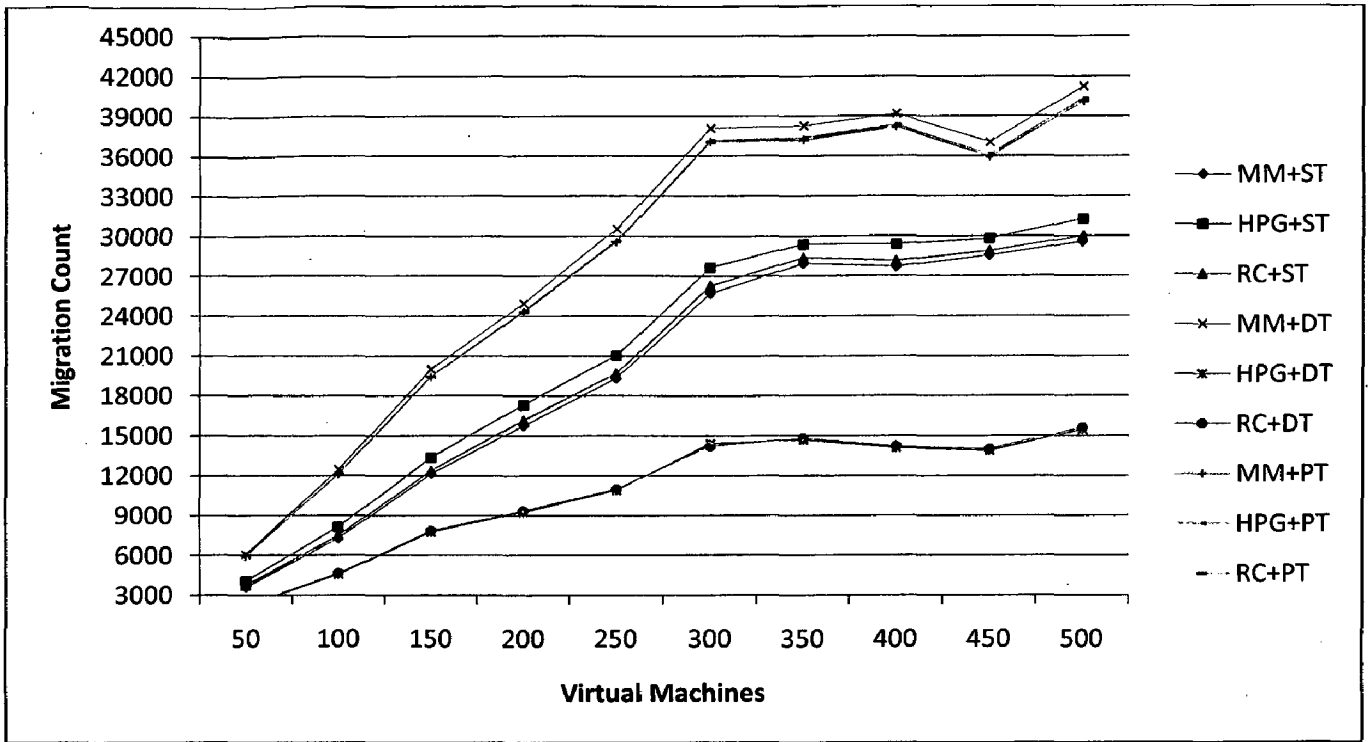


Figure 5.2. : The variation of virtual machine migration count with virtual machine count taking 100 hosts

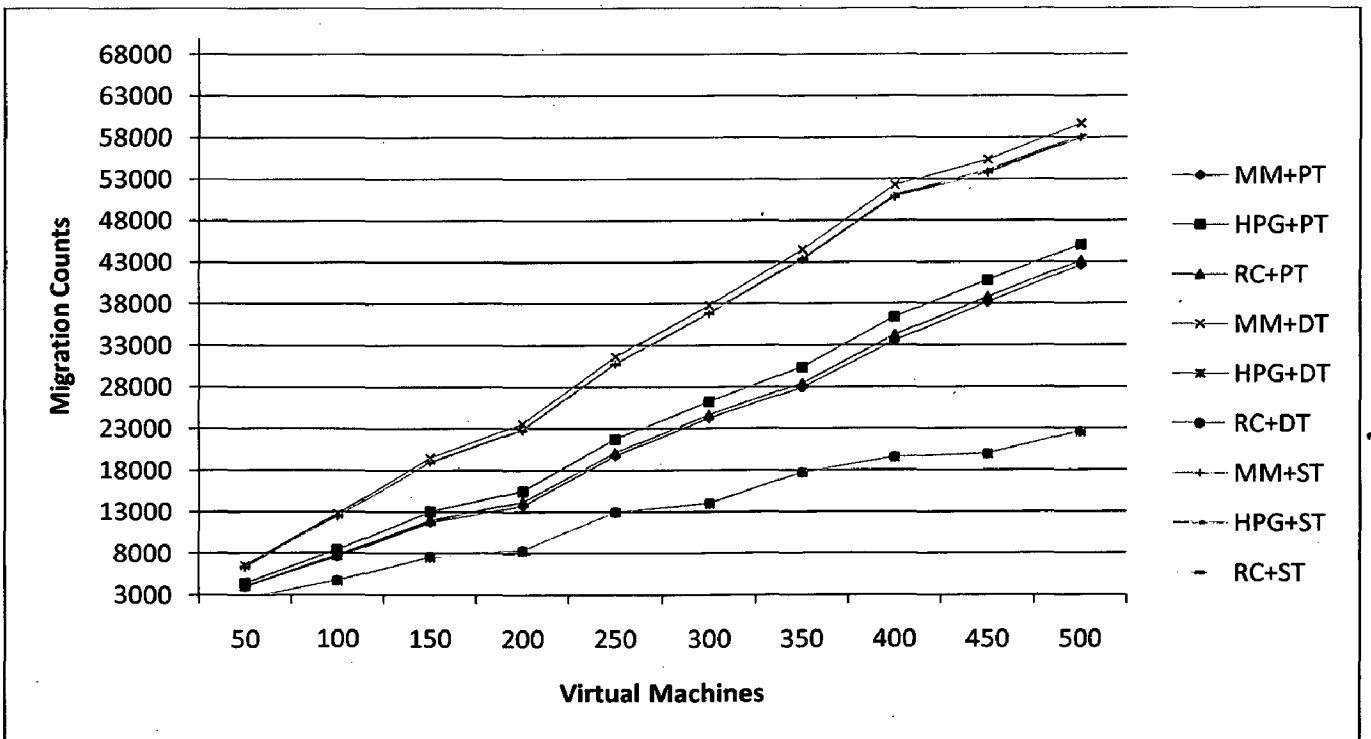


Figure 5.3. : The variation of virtual machine migration count with virtual machine count taking 150 hosts

The figure 5.3 shows the variation of migration count with the virtual machines. The simulation is done taking 150 hosts. The MM policy using proposed threshold schemes performs best giving a lower migration count. The MM policy using the DT policy gives the highest migration count. The overall migration count increases as the virtual machine count increases. The graph also shows linear increase in the migration count with respect to the virtual machine count as at 150 hosts all the virtual machines are allocated to some host or the other and is getting the full specification as expected.

The figure 5.4 shows the variation of migration count with the virtual machines. The simulation is done taking 200 hosts. The MM policy using proposed threshold schemes performs best giving a lower migration count. The RC policy using the proposed threshold gives the highest migration count. The sudden increase in the migration count for the MM using ST policy is because the host utilization varies as the number of virtual machines allocated to it increases, but in the ST policy the threshold values are given as constant, thus increasing the migration of virtual machines.

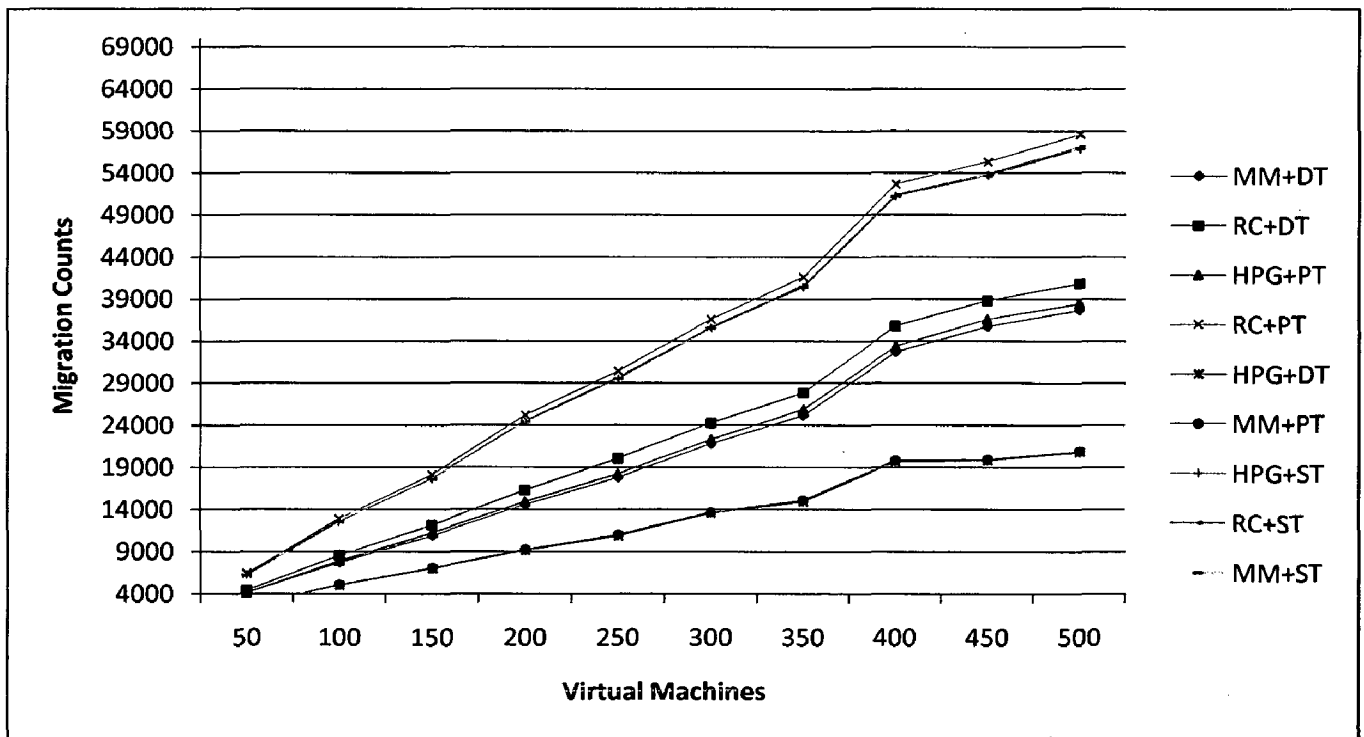


Figure 5.4. : The variation of virtual machine migration count with virtual machine count taking 200 hosts

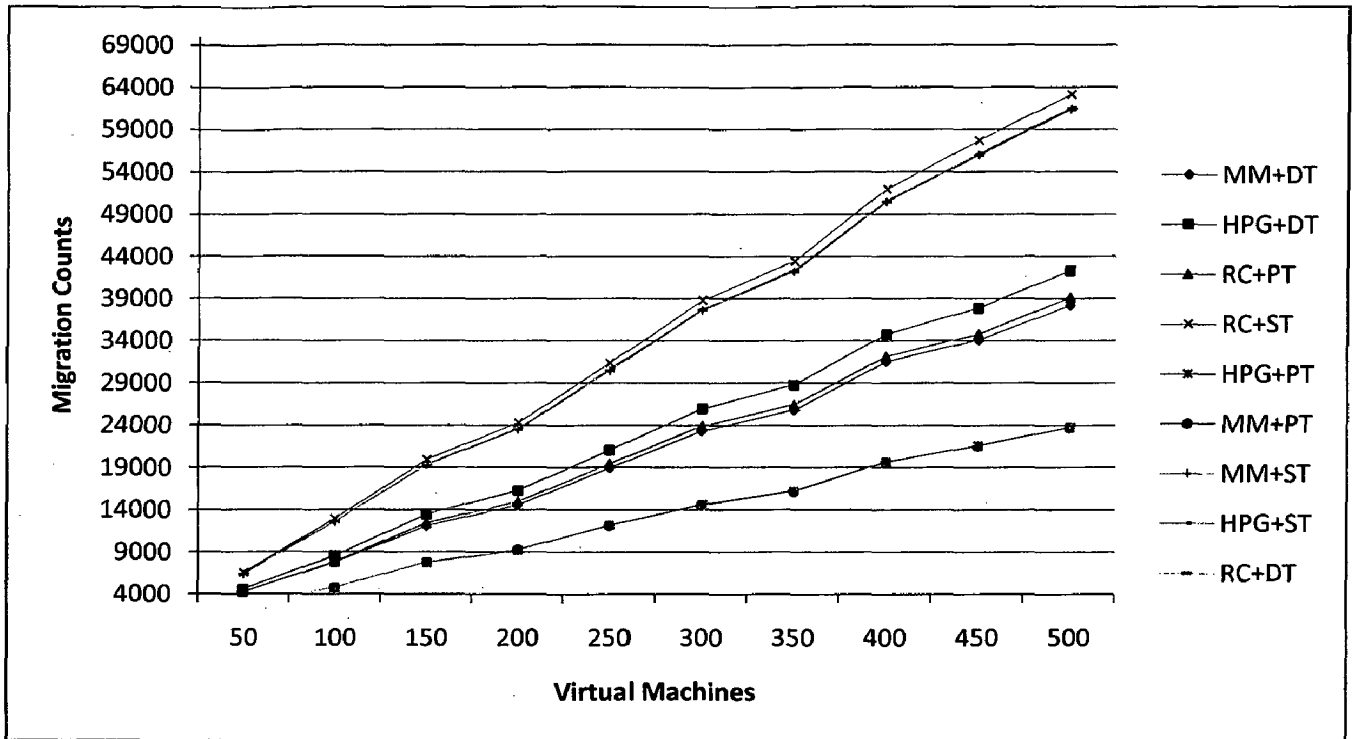


Figure 5.5. : The variation of virtual machine migration count with virtual machine count

The figure 5.5 shows the variation of migration count with the virtual machines. The simulation is done taking 250 hosts. The MM policy using proposed threshold schemes gives similar result to the HPG policy using proposed threshold. The RC policy using the static threshold gives the highest migration count. The graph is linear as all the host is 250 most of the virtual machines are allocated. There is very less difference seen in some of the lines. This is because the difference is insignificant, that is a difference of maximum 500 is there which is not visible in comparison to the higher values of the migration count.

The figure 5.6 shows the variation of migration count with the virtual machines. The simulation is done taking 300 hosts. The MM policy using proposed threshold schemes performs best. The MM policy using the ST gives the highest migration count. The result is almost similar to the previous graph.

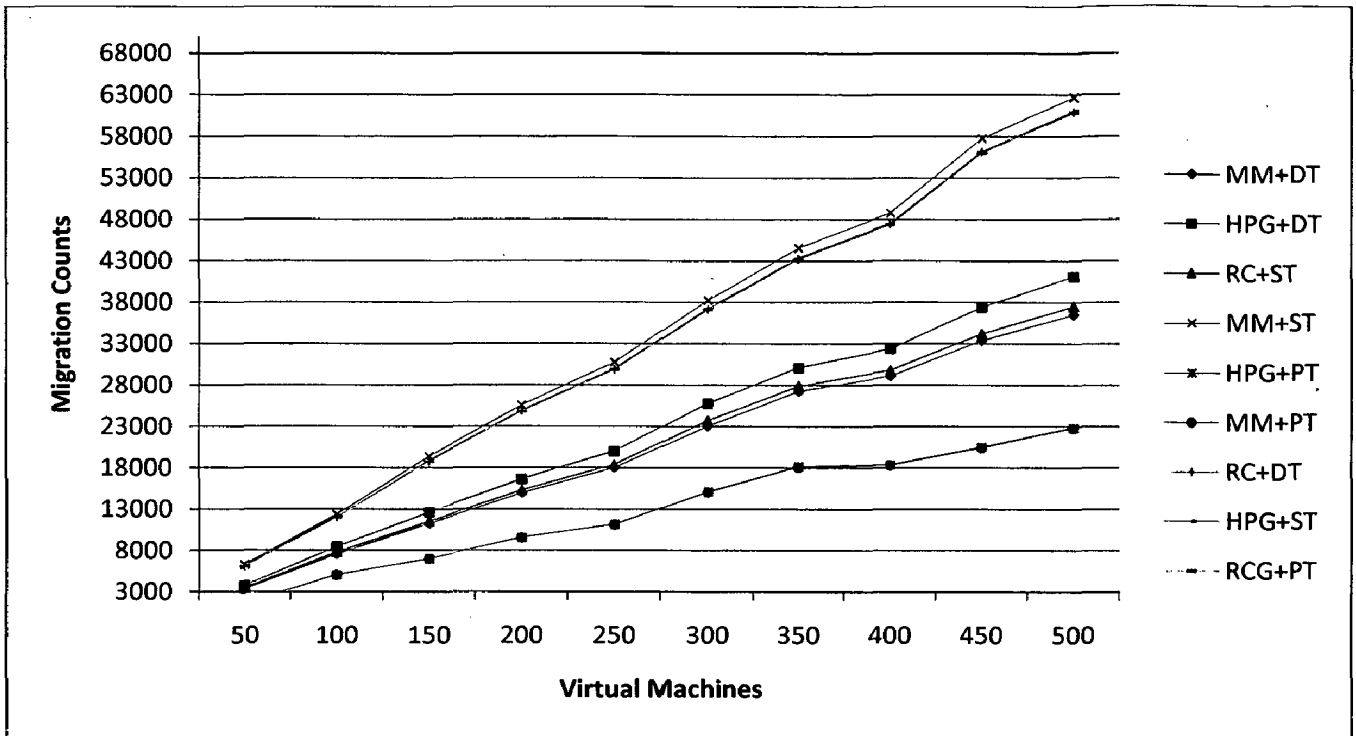


Figure 5.6. : The variation of virtual machine migration count with virtual machine count taking 300 hosts

In real time application were there are large number of host and virtual machines the dynamic threshold policy can be used. The static threshold policy gives large number of migration count virtual machines in comparison to the dynamic threshold policy. For smaller number of host and large number virtual machines the migration count of virtual machine decreases because of the fact that all the virtual machines can't be allocated to the host as host specification does not support them

5.2. Evaluation of the Migration Policies using Proposed Threshold Policy

In this section the result of the evaluation done for the different migration policies using dynamic threshold policy is shown.

The different migration policies are:

1. Modified Roulette Wheel Selection Policy
2. Minimization of Migration Policy
3. Highest Potential Growth Policy

4. Random Choice Policy

The figure 5.7, gives the variation of different migration policies using proposed threshold scheme for 50 hosts. The graph shows that there is very slight distinction between the migration policies. The difference is in the order of 100 which are not clearly visible as the migration count is in the order of 10000. There are various dips seen in the graph. It is due to the fact that taking 50 hosts many of the virtual machines are not allocated as the specifications are not met.

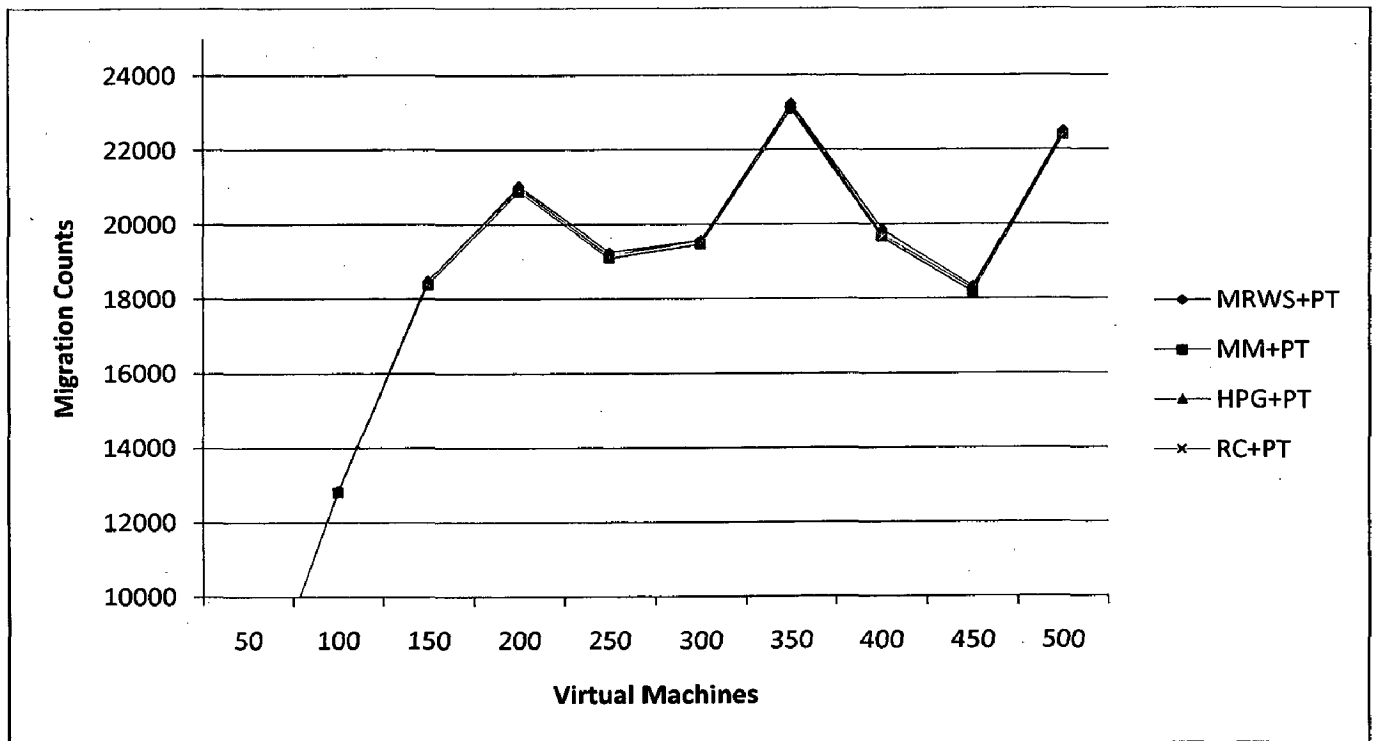


Figure 5.7. : The variation of virtual machine migration count with virtual machine count taking 50 hosts

The figure 5.8, gives the variation of different migration policies using proposed threshold scheme for 100 hosts. The graph is linear till 300 virtual machines. After that the graph is horizontal with a dip. It is due to the reason that for 100 hosts 300 virtual machines can be assigned. As the number of virtual machines increases, they are not allocated to the hosts. A dip is the exhaustion limit of the 100 hosts.

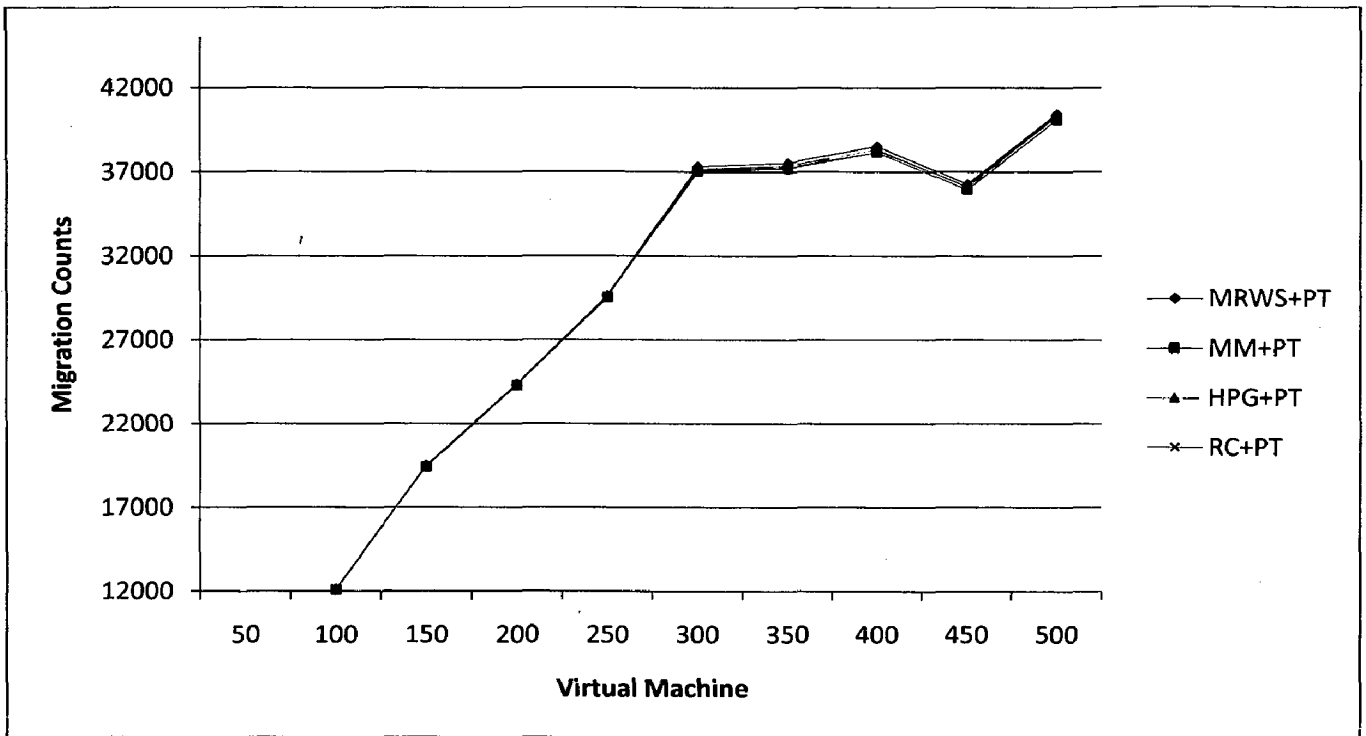


Figure 5.8. : The variation of virtual machine migration count with virtual machine count taking 100 hosts

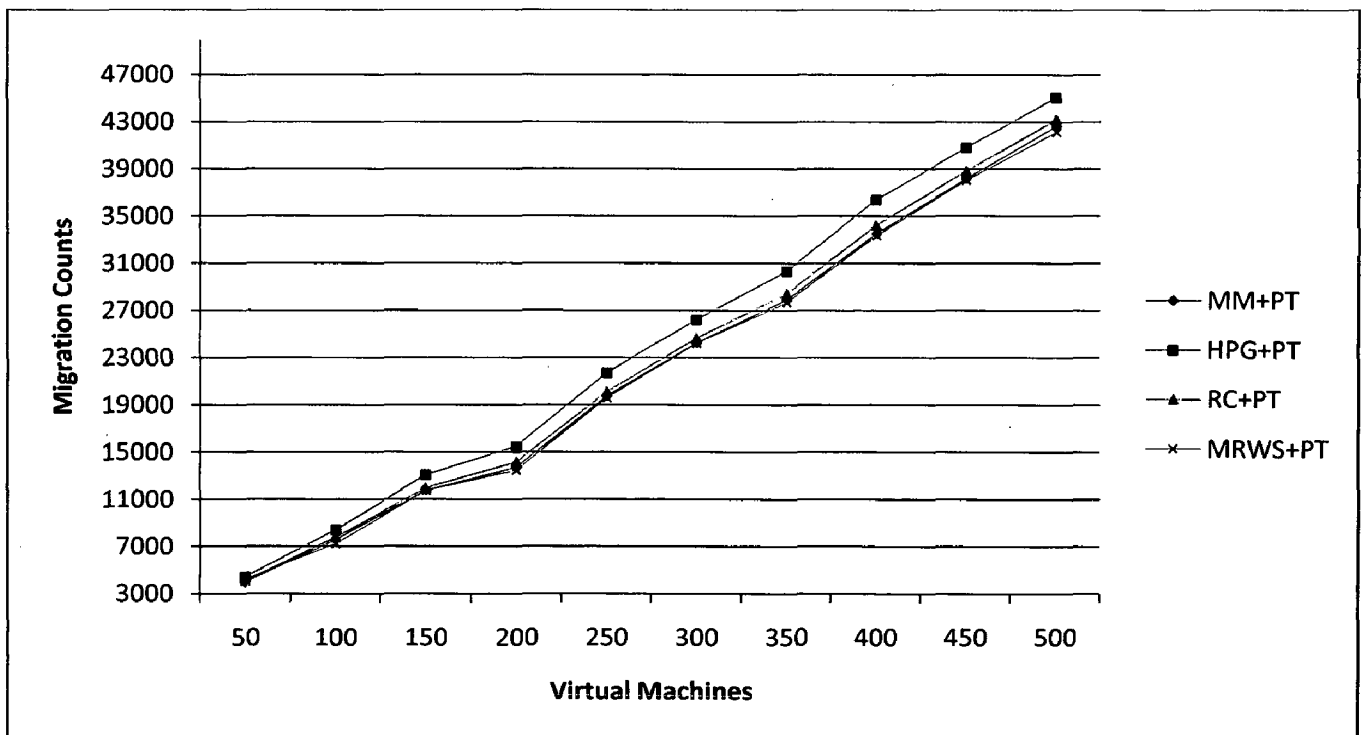


Figure 5.9. : The variation of virtual machine migration count with virtual machine count taking 150 hosts

The figure 5.9, gives the variation of different migration policies using proposed threshold scheme for 150 hosts. The graph shows that MRWS performs best and HPG gives the worst performance. The graph follows linear path. All the virtual machines are fully allocated to the hosts. The difference is significant between the HPG and the rest of the policies.

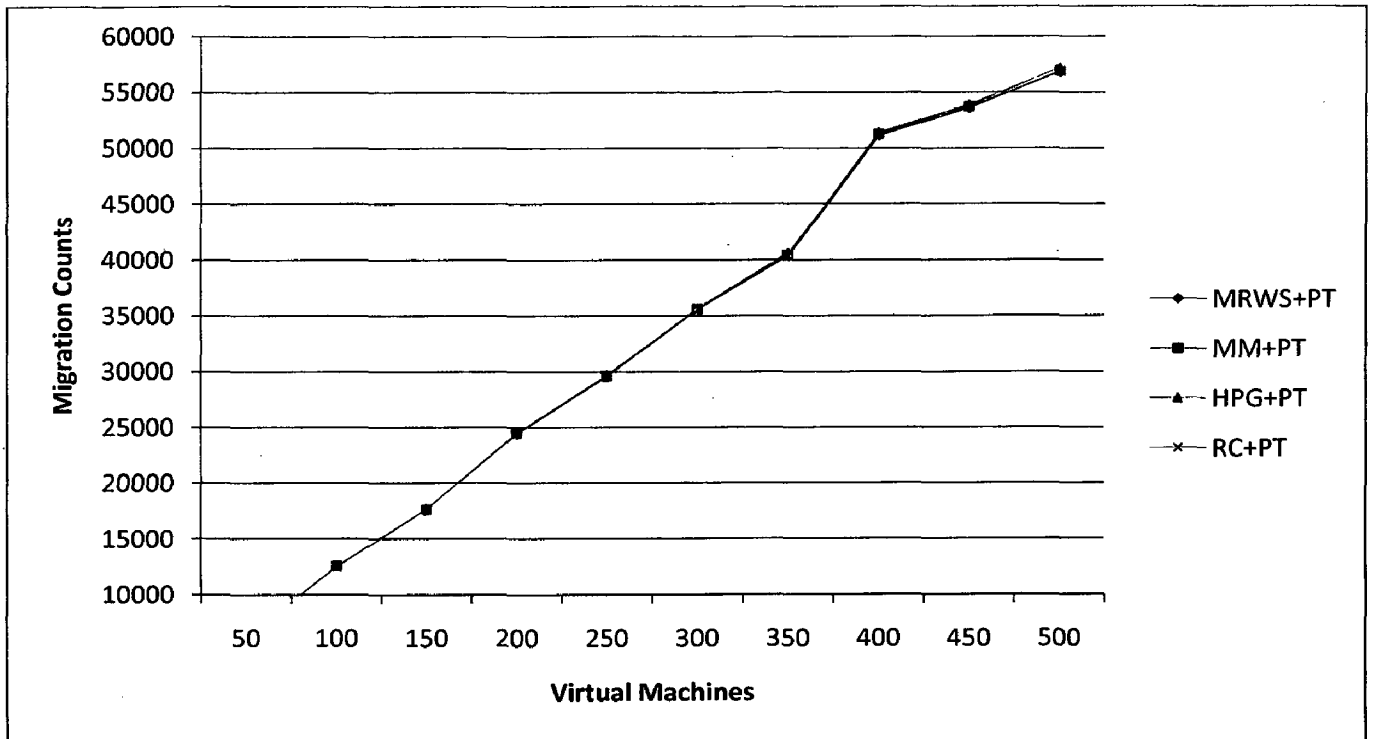


Figure 5.10. : The variation of virtual machine migration count with virtual machine count taking 200 hosts

The figure 5.10, gives the variation of different migration policies using proposed threshold scheme for 200 hosts. The graph shows that there very slight distinction between the migration policies that is of the order of 100 which is negligible in comparison to the migration count which is in the order of 1000.

The figure 5.11, gives the variation of different migration policies using proposed threshold scheme for 250 hosts. The graph shows that there very slight distinction between the migration policies which is similar to the previous graph.

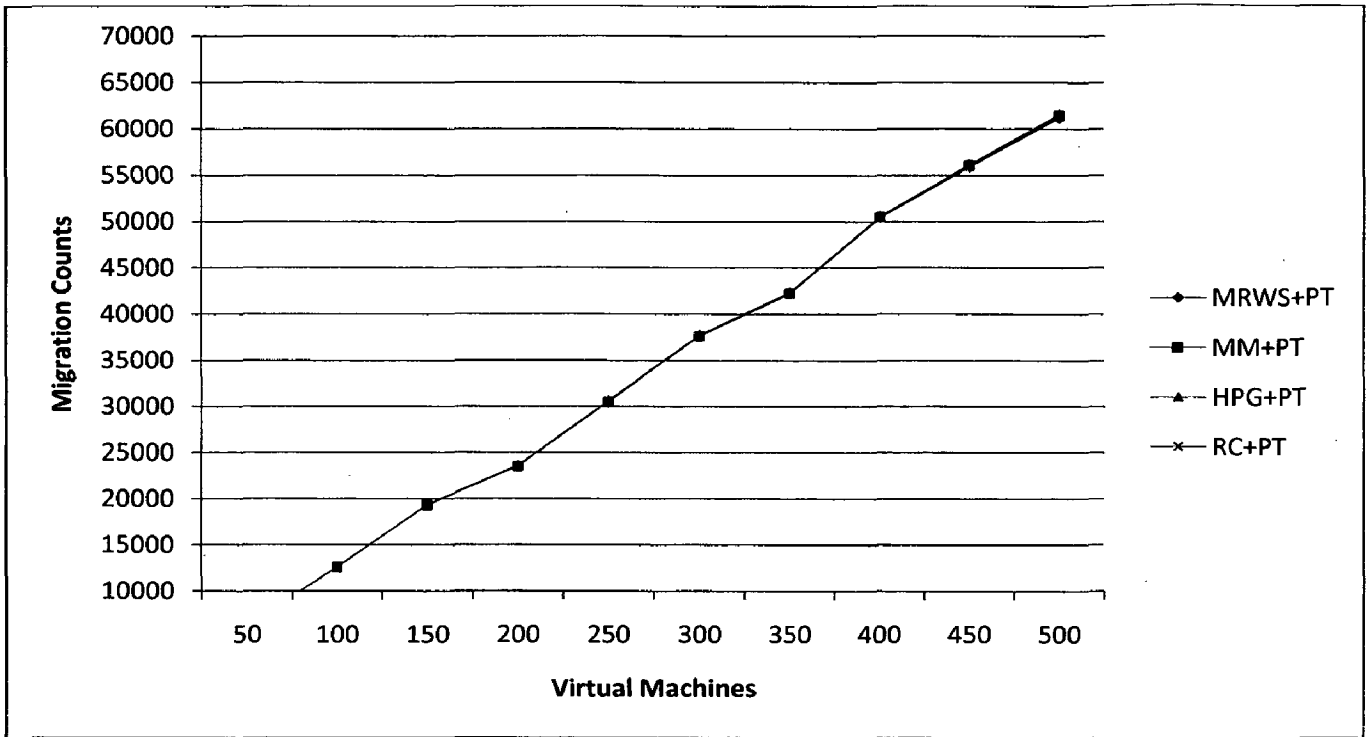


Figure 5.11. : The variation of virtual machine migration count with virtual machine count taking 250 hosts

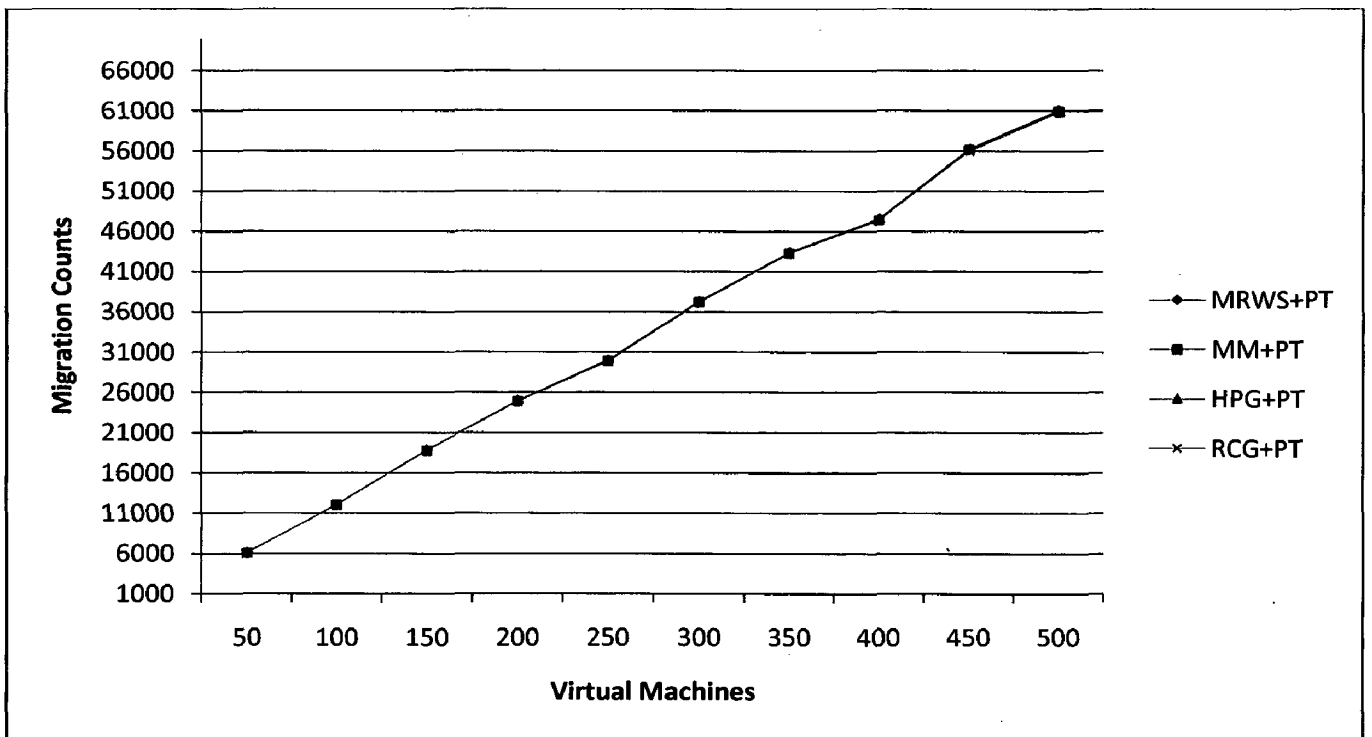


Figure 5.12. : The variation of virtual machine migration count with virtual machine count taking 300 hosts

The figure 5.12, gives the variation of different migration policies using proposed threshold scheme for 300 hosts. The graph shows that there is very distinction between the migration policies. The graph is similar to the previous 2 graphs, thus showing the results are consistent.

As shown in the figures 5.7- 5.12, it is found that as the number of host increases, there is a decrease in the migration count of the virtual machine in the Modified Roulette Wheel Selection policy. There is no significant difference in the migration count of the migration policies. But in the real time applications where there are number of large number of hosts and the virtual machines, the difference in the migration count would be observed. As shown in the figures as number of host increases a difference is noticed. It is also clear that for smaller number of host and large number virtual machines the migration count of virtual machine decreases. It is because of the fact that all the virtual machines can't be allocated to the host as host specification does not support them.

5.3. Evaluation of the Proposed Threshold Policy

The proposed threshold scheme takes into account the previous threshold values and the current utilization of the host. The initial value of the lower and upper threshold values is taken as a and b times the initial host utilization. In this section the variation of the migration count for virtual machines with respect to the constants (a) is discussed. The simulation is done taking 100 host and 500 virtual machines. The sum of a and b is 1. Thus the value b is depended upon the value of a .

The figure 5.13. shows the variation of lower and upper utilization threshold averaged over 100 hosts with the a values. The value of a varies from 0 to 0.5. The value of a cannot be greater than 0.5 because in that case the value of lower utilization threshold would be greater than upper utilization threshold, which is not possible. As seen from the graph the average value of utilization thresholds is very low initially. It is because of the fact that at the start of the simulation most of the virtual machines are not allocated to the host. This average value changes from around 10 MIPS to a maximum 220 MIPS. As seen from the graph, the

maximum value of utilization is around 0.3 a value. The difference between the lower and upper threshold values is also greater in case of 0.3. The higher difference indicates more probability of host utilization to lie in this range, thus decreasing the virtual machine migration count

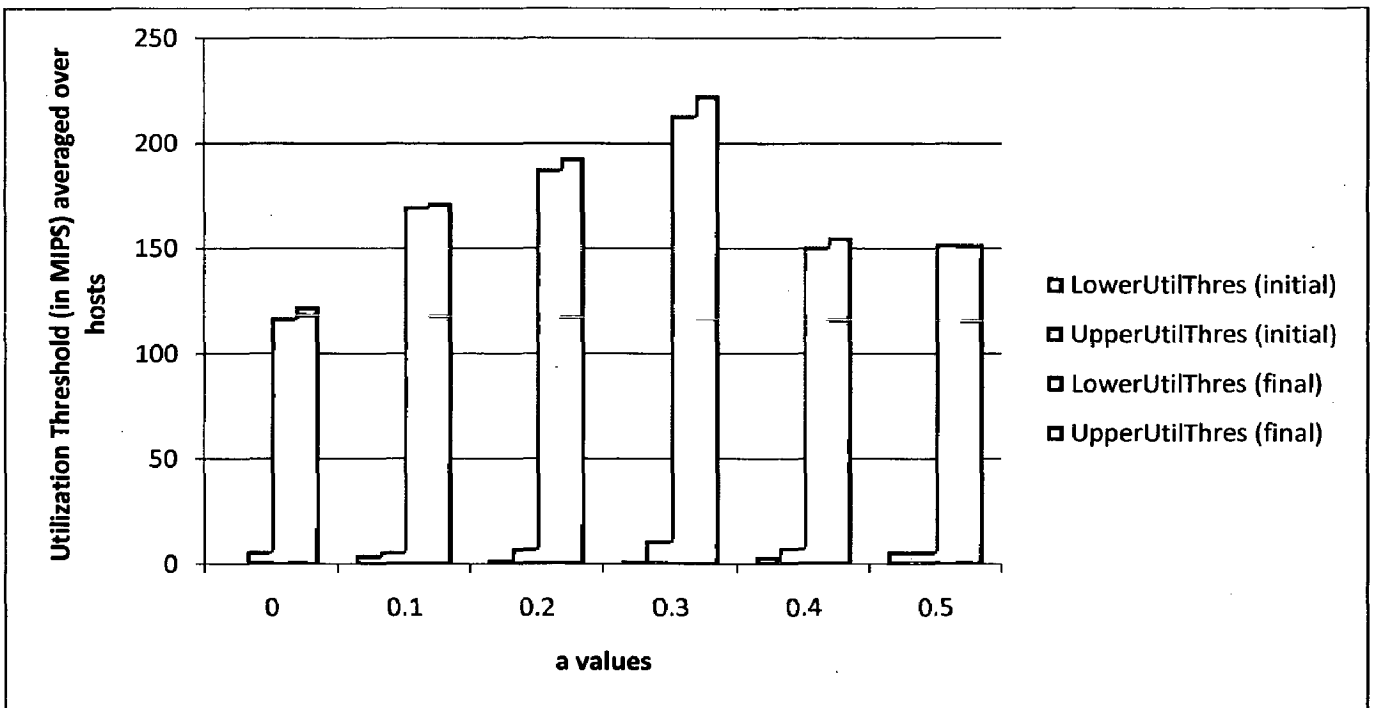


Figure 5.13. : The variation of utilization threshold averaged over 100 hosts with the a values

CHAPTER 6

CONCLUSION

Cloud Computing is fast spreading technologies. It is helpful in providing a large number of services of the user. User can share data and information and can retrieve them. The user can also use the application hosted on various datacenter. There are large numbers of datacenter which are distributed geographically. These datacenter consumes large amount of energy which need to be reduced. As decreasing the energy consumption will not only save the energy but also reduce the cost due to energy consumption. There are large numbers of hosts and different virtual machines are allocated to these host. The number of host whose utilization is below a given a threshold level can be shut down and the virtual machines on that host can be switched to other host.

In this dissertation, cloud computing is discussed including various types of cloud and different deployment methods. The different migration policy along with the dynamic threshold evaluation scheme is discussed in detail. The modification of Roulette Wheel Selection policy for virtual machine migration is suggested. A dynamic threshold evaluation scheme for the determining the threshold is suggested. The CloudSim, the simulation environment for the cloud is discussed in detail.

The dissertation gives the performance evaluation in terms of migration count of virtual machines. The evaluation is done for the three migration policy using the static threshold and dynamic threshold policies. The result indicates that the migration policy performs better in case of the proposed dynamic threshold evaluation policy and the migration policies give worst performance incase of static threshold values. The static threshold values cannot be used in real time scenarios where the workload changes.

The suggested migration policy is compared with the given migration policies using the proposed dynamic threshold policy. The results indicated that the performance is better for the proposed migration policy, although the difference is very less. The results are consistent with varying number of hosts. Thus for larger number of hosts the proposed migration

scheme gives better results in comparison to the other migration policies using proposed dynamic threshold policy.

The evaluation of the proposed dynamic threshold policy is done. The variation of lower and upper utilization threshold averaged over hosts with the α values, indicates that the possible values of α can lie between 0 and 0.5. The difference in the threshold values indicates that the optimal α value is around 0.3.

6.1. Suggestion for future work

1. The future work is to use the migration list generated to effectively migrate the virtual machine to different host and to minimize the energy consumption.
2. The allotment of the virtual machines to the host can be modified so that after the allotment the host utilization is between the two threshold values.
3. The simulation was done with host having one CPU core. The simulation has to be done on multi-core host as well.

REFERENCES

- [1]. Gowrigolla B., Sivaji, S., Masillamani M.R., "Design and auditing of Cloud computing security," Proc. Information and Automation for Sustainability (ICIAFs), 20105th International Conference, Dec. 2010, pp.292-297.
- [2]. Anton Beloglazov, JemalAbawajy, and RajkumarBuyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing", The International Journal of Grid Computing and eScience, Future Generation Computer Systems (FGCS), , Apr 2011, vol. 28, no. 5, pp. 755-768.
- [3]. Ahmed, M.; Yang Xiang; Ali, S., "Above the Trust and Security in Cloud Computing: A Notion Towards Innovation," Proc. Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference, Dec 2010, pp. 723-730.
- [4]. Almulla, S.A., Chan YeobYeun, "Cloud computing security management," Proc. Engineering Systems Management and Its Applications (ICESMA), 2010 Second International Conference, Apr 2010, pp.1-7.
- [5]. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and RajkumarBuyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Wiley Online Library, Aug 2010, vol. 41, no.1, pp. 23- 50.
- [6]. RichaSinha, NidhiPurohit and HiteshiDiwanji, "Power Aware Live Migration for Data Centers in Cloud using Dynamic Threshold", International Journal of Computer Technology and Applications, Dec 2011, vol. 2, no. 6, pp. 2041-2046.
- [7]. S. Rajasekaran, G.A. VijayalakshmiPai, "Neural Network, Fuzzy Logic, and Genetic Algorithms", PHI Learning pvt. ltd., May 2010

- [8]. Xiaobo Fan , Wolf-Dietrich Weber , Luiz André Barroso. "Power provisioning for a warehouse-sized computer". In Proc. of the 34th ACM International Symposium. On Computer Architecture, May 2007, vol. 35, no. 2, pp. 13-23.

- [9]. Akshat Verma, Puneet Ahuja, Anindya Neogi "pMapper: power and migration cost aware application placement in virtualized systems". In Proc. of the 9th ACM/IFIP/USENIX International Conference on Middleware 2008, pp. 243-264.

- [10]. Elisa Bertino, Federica Paci, Rodolfo Ferrini, Ning Shang, "Privacy-preserving Digital Identity Management for Cloud Computing". IEEE Data Engineering Bulletin, Mar 2009, vol. 32, no. 1, pp. 21-27.

- [11]. Buyya, R.; Chee Shin Yeo; Venugopal, S., "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference, Sep 2008 pp.5-13.