Multiple Correlation Coefficient Approach for VM Migration

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Sabina Singh
(Roll No.801032032)

Under the supervision of:
Ms. Anju Bala
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Jan-2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "Multiple Correlation Coefficient Approach for VM Migration", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Anju Bala and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:
(Sabina Singh)

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

(Anju Bala)
Assistant Professor, CSE

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

The thesis on “Multiple Correlation Coefficient Approach for VM Migration” has been of great personal importance for me and this has been a learning experience that could not have been possible without the support of my guide Mrs. Anju Bala., Assistant Professor.
I would also like to thank Dr. Maninder Singh, Head of Computer Science & Engineering Department, for their immense support and help extended throughout this time.
I would like to thank my family for being there when I needed them the most, my friends who stood by me through the thick and thin.
I would like to acknowledge that without your support, this would have been very difficult indeed.

Sabina Singh
(801032032)

Abstract

Cloud computing represents a major step up in computing whereby shared computation resources are provided on demand. In such a scenario, applications and data therefore, can be hosted by various virtual machines (VMs). Therefore, placement of virtual machines which host an application and migration of these virtual machines while the unexpected network latency or congestion occurs is critical to achieve and maintain the application performance. To address these issues, new VM migration approach is proposed which is an extension of the maximum correlation analysis. The previous approach was working on Maximum Correlation strategy to find which VMs have some high association based on only single criteria. However, such consolidation cannot trivial in sense that migration of VM can be based on only two factors, as it can result in violations of the SLA negotiated with customers and other current factors that playing role that time in degrading the current performance of the datacenter. Therefore, multiple maximum correlation approach is proposed which will be able to select faster based on the correlation of multiple factors. The correlation matrix is used which is based on multiple factors for efficient dynamic consolidation and migration of VMs due to some adversity or over utilization of resources. To maximize their reliability of their datacenters the Cloud providers have to apply VM migration policy.

Table of Contents

List of Figures

List of Tables

Chapter-1

Introduction

1.1 Cloud Computing

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The datacenter hardware and software called as a Cloud [20]. Cloud computing is an IT deployment model, based on virtualization, where resources, in terms of infrastructure, applications and data are deployed via the internet as a distributed service by one or several service providers. These services are scalable on demand and can be priced on a pay-per-use basis [1].

Cloud computing, like much of IT, builds on earlier technologies and concepts, e.g., utility computing, grids, service oriented architecture, and virtualization [2]. In computer science by 'cloud' is understood a network of computing devices which work together to provide services. More specifically, in web hosting cloud computing means that all web hosting services run on many different servers ensuring that a failure in one device will not cause a service failure [3]. Cloud computing received significant attention recently as it changes the way computation and services to customers, For example, it changes the way of providing and managing computing resources, such as CPUs, databases, and storage systems.[4]

In essence, cloud computing is a construct that allows you to access applications that actually reside at a location other than your computer or other internet –connected device ;most often , this will be a distant datacenter. Cloud computing promises to cut operational and capital costs and, more importantly, let IT departments focus on strategic projects instead of keeping the datacenter running. [5]

1.2 Cloud Components

- Global Manager – A component that is deployed on the management host and makes global management decisions, such as mapping VM instances on hosts, and initiating VM migrations.

- Local Manager – A component that is deployed on every compute host and makes local decisions, such as deciding that the host is under loaded or overloaded and selecting VMs to migrate to other hosts.
- Data Collector – A component that is deployed on every compute host and is responsible for collecting data about the resource usage by VM instances, as well as storing these data locally and submitting the data to the central database [10]

1.3 Elements of a cloud computing solution-

A cloud computing solution has several elements that exhibits specific role in delivering a functional cloud based application. They are clients, the datacenter, and distributed servers.

- Clients- They is the devices to which the end users interact for managing their information on the cloud. Clients fall in three categories, which are: mobile, thin, thick.
- Mobile- PDAs or smart phones.
- Thin- These clients are the computers that are without hard drives, but rather allow the servers do all the work, but display the information.
- Thick-These are the clients that are the general computers which makes use of Web browsers to connect to the cloud
- Datacenter- It is the collection of servers where the application to which you subscribe is housed. [5]
- Distributed Servers- It is not necessary that the servers need to be housed at the same location, rather servers are in geographically different locations.

1.4 Service Models

The characteristics of cloud computing are offered in different service models, these models deliver their services to different types of users. In the figure shown below the type of user per service model is shown. The service models are also called the cloud stack.

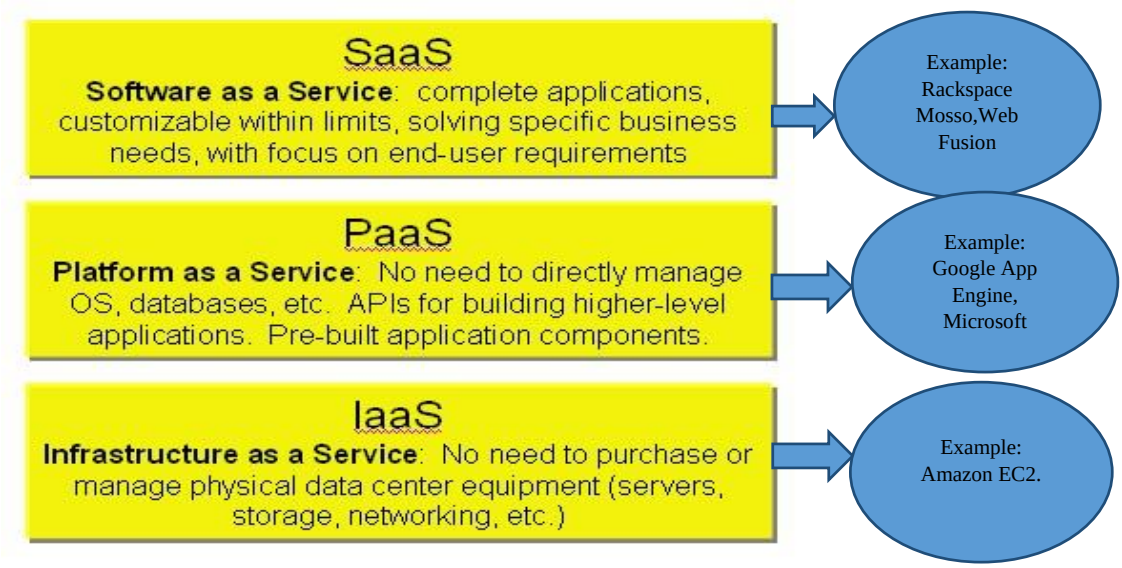

Figure 1.1 –Service Models

1.4.1 Software as a Service (SaaS)

Software as a Service (SaaS) is cloud computing layer where users simply make use of a web-browser to access software that others have developed, maintain and offer as a service over the web. At the SaaS level, users do not have control or access to the underlying platform and infrastructure that is being used to host the software. Salesforce's Customer Relationship Management and Google gmail are popular examples that use the SaaS model of cloud computing.[20].

1.4.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) is the layer where applications are developed using a set of programming languages and tools that are supported and provided by the PaaS provider. PaaS provides developers with a high level of abstraction that allows them to focus on developing their applications. Developers can provide their customers with an custom developed application without the hassle of defining and maintaining the infrastructure. Just like the SaaS model, users do not have control or access to the underlying infrastructure being used to host their applications at the PaaS level. Google App Engine and Microsoft Azure are popular PaaS examples [7].

1.4.3 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) is the lowest layer where users use computing resources such as databases, CPU power, memory and storage from an IaaS provider and use the

resources to deploy and run their applications. In contrast to the PaaS model, the IaaS model allows users to access the underlying infrastructure through the use of virtual machines which automatically can scale up and down. IaaS gives users more flexibility than PaaS as it allows the user to deploy any software stack on top of the operating system. However, flexibility comes with a cost and users are responsible for updating and patching the operating system at the IaaS level. Amazon Web Services' EC2 and S3 are popular IaaS examples [8].

1.5 Deployment Models

1.5.1 Private cloud

The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise. This model doesn't bring much in terms of cost efficiency: it is comparable to buying, building and managing your own infrastructure. Still, it brings in tremendous value from a security point of view. During their initial adaptation to the cloud, many organizations face challenges and have concerns related to data security. These concerns are taken care of by this model, in which hosting is built and maintained for a specific client. The infrastructure required for hosting can be on-premises or at a third-party location [9]

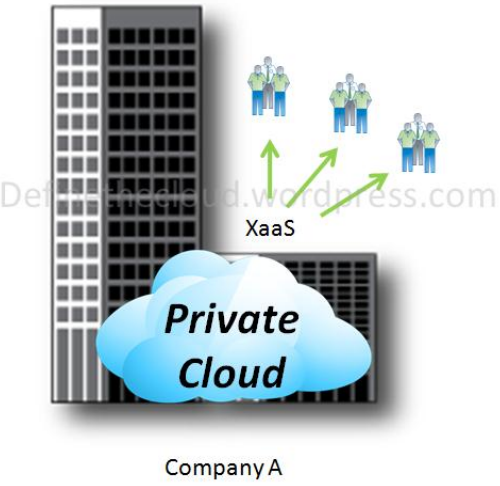

Figure 1.2-Private Cloud [6]

1.5.2 Community cloud

The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and

compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise. In the community deployment model, the cloud infrastructure is shared by several organizations with the same policy and compliance considerations. This helps to further reduce costs as compared to a private cloud, as it is shared by larger group [8].

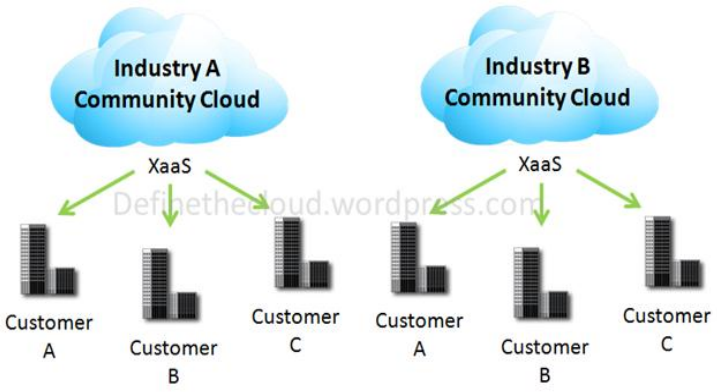

Figure 1.3- Community Cloud [6]

1.5.3 Public cloud

The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services. In this deployment model, services and infrastructure are provided to various clients. Google is an example of a public cloud. This service can be provided by a vendor free of charge or on the basis of a pay-per-user license policy. This model is best suited for business requirements. This model helps to reduce capital expenditure and bring down operational IT costs [8].

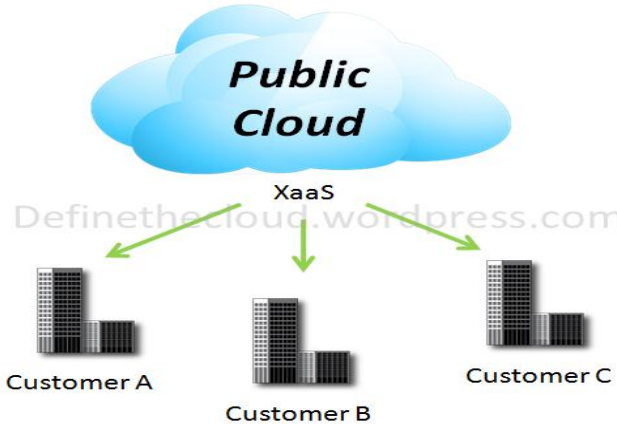

Figure 1.4- Public Cloud [6]

1.5.4 Hybrid cloud

The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. This deployment model helps businesses to take advantage of secured applications and data hosting on a private cloud, while still enjoying cost benefits by keeping shared data and applications on the public cloud. This model is also used for handling cloud bursting, which refers to a scenario where the existing private cloud infrastructure is not able to handle load spikes and requires a fallback option to support the load. Hence, the cloud migrates workloads between public and private hosting without any inconvenience to the users [8].

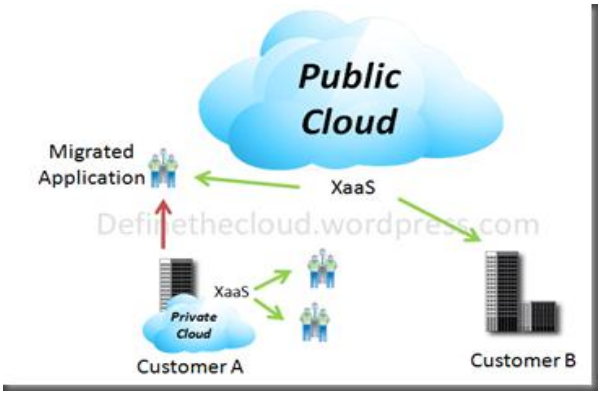

Figure 1.5- Hybrid Cloud [6]

1.6 Benefits of Cloud Computing

Cloud computing offers various services as listed below:

On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources

dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

Rapid elasticity: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Measured Service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled and reported providing transparency for both the provider and consumer of the utilized service.

Chapter 2

Literature Survey

2.1 Fault Tolerance

Fault-tolerant computing is the art and science of building computing systems that continue to operate satisfactorily in the presence of faults the ability for a system to remain in operation even if some of the components used to build the system fail [11]. Fault tolerance is a main concern for the assurance of availability and reliability of critical services as well as application execution. To minimize failure impact on the system and application execution, failures must be predicted and proactively handled. In some cases, if fault occurs, a reactive measure needs to be implemented to protect the system from its consequences Fault tolerance techniques are used to predict these failures and take an appropriate action before failures actually occur.

Examples of systems in which fault tolerance is needed include mission-critical, computation-intensive, transactions (such as banking), and mobile/wireless computing systems/networks. High performance, measured in terms of speed and computing power, is essentially used as major design objective for such systems. It is however conceivable that great loss of crucial transactions can take place due to a small system/component error. The emergence of new paradigms, such as mobile/wireless computing, requires the introduction of new techniques for fault tolerance. It is therefore prudent that the issue of fault tolerance become among the set of design objectives of current and future computing systems [12].

Fault tolerance makes a system capable to operate correctly in a faulty condition, protect against accidental or malicious destruction of information, protect against generating erroneous output and guarantees that confidential information cannot be divulged. Fault Tolerance is one of the key issues of cloud computing. Fault tolerance is concerned with all the techniques necessary to enable a system to tolerate software faults remaining in the system after its development. These software faults may or may not manifest themselves during systems operations, but when they do, software fault tolerant techniques should provide the necessary mechanisms of the software system to prevent system failure

occurrences. With increasing heterogeneity and complexity of computational clouds, and due to the inherent unreliable nature of large-scale cloud infrastructure, fault tolerance techniques have become a major concern. Fault tolerance techniques are designed to allow a system to tolerate software faults that remain in the system after its development. Fault tolerance techniques are employed during the procurement, or development, of the software. When a fault occurs, these techniques provide mechanisms to the software system to prevent system failure from occurring [13]. It should also be incorporated in autonomic cloud computing environment.

2.2 Basic Concepts

2.2.1 Concepts and Terminologies

- **Dependability**

Being fault tolerant is strongly related to what are called dependable systems. Dependability is a term that covers a number of useful requirements for distributed systems including the following:

- **Availability**

Availability is defined as the property that a system is ready to be used immediately. In general, it refers to the probability that the system is operating correctly at any given moment and is available to perform its functions on behalf of its users.

- **Reliability**

Reliability refers to the property that a system can run continuously without failure. A highly reliable system is one that will most likely continue to work without interruption during a relatively long period of time.

- **Safety**

Safety refers to the situation that when a system temporarily fails to operate correctly, nothing catastrophic happens.

- **Maintainability**

Maintainability refers to how easy a failed system can be repaired. A highly maintainable system may also show a high degree of availability, especially if failures can be detected and repaired automatically.

- **Fault**

Fault can be defined as incorrect state of hardware or software resulting from physical defects, design flaw or operator error.

Faults can be classified into one of three categories:

- Transient faults
- Intermittent faults
- Permanent faults

Transient faults occur once and then disappear. For example, a network message transmission times out but works fine when attempted a second time. Intermittent faults are the most annoying of component faults. This fault is characterized by a fault occurring, then vanishing again, then occurring. An example of this kind of fault is a loose connection. Permanent faults are persistent: it continues to exist until the faulty component is repaired or replaced. Examples of this fault are disk head crashes, software bugs, and burnt-out hardware.

- **Error**

Error is a part of a system state that may lead to a failure or we can say error is the manifestation of a fault.

- **Failure**

When a system or module is designed, its behavior is specified. When observed behavior differs from the specified behavior we call it as a failure.

Failures can be classified as:

- Crash failure
- Transient failure
- Byzantine failure
- Temporal failure

A process undergoes crash failure, when it permanently ceases to execute its actions. This is an irreversible change. The agent inducing transient failure may be temporarily active, but it can make a lasting effect on the global state. When a process behaves arbitrarily, we say that Byzantine failure has occurred. Real time systems require actions to be completed within a specific amount of time. When the deadline is not met, a temporal failure occurs.

Mechanisms are added to detect errors and implement recovery.

2.2.2 Hardware Fault-Tolerance

Two general approaches to hardware fault recovery have been used:-

- Fault masking,
- Dynamic recovery.

2.2.2.1 Fault masking

It is a structural redundancy technique that completely masks faults within a set of redundant modules. A number of identical modules execute the same functions, and their outputs are voted to remove errors created by a faulty module interruption when a fault occurs, allowing existing operating systems to be used.

2.2.2.2 Dynamic recovery

It is required when only one copy of a computation is running at a time (or in some cases two unchecked copies), and it involves automated self-repair. As in fault masking, the computing system is partitioned into modules backed up by spares as protective redundancy. In the case of dynamic recovery however, special mechanisms are required to detect faults in the modules, switch out a faulty module, switch in a spare, and instigate those software actions (rollback, initialization, retry, and restart) necessary to restore and continue the computation. In single computers special hardware is required along with software to do this, while in multi-computers the function is often managed by the other processors. Dynamic recovery is generally more hardware-efficient than voted systems, and it is therefore the approach of choice in resource-constrained (e.g., low-power) systems, and especially in high performance scalable systems in which the amount of hardware resources devoted to active computing must be maximized. Its disadvantage is that computational delays occur during fault recovery, fault coverage is often lower, and specialized operating systems may be required.

2.2.3 Software Fault-Tolerance

Efforts to attain software that can tolerate software design faults (programming errors) have made use of static and dynamic redundancy approaches similar to those used for hardware faults [11].

2.3 Fault Tolerance policies

There are two major fault tolerance policies -

2.3.1 Reactive Fault Tolerance

The rate of failures in larger systems is bound to increase, that because reactive schemes to result in increased I/O bandwidth requirements, which can become a bottleneck [14]. Reactive Fault Tolerance keeps parallel applications alive through recovery from experienced failures. Reactive Fault Tolerance keeps parallel applications alive through recovery from experienced failures. It is used to diminish the influence of failures on application execution when the failure occurs. Reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. There are various techniques which are based on these policies like Checkpoint/Restart, Replay and Retry and so on.

- Check pointing/ Restart - When a task fails, it is allowed to be restarted from the recently checked pointed state rather than from the beginning. It is an efficient task level fault tolerance technique for long running applications.
- Replication-Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. It can be implemented using tools like HAProxy, Hadoop and AmazonEc2 etc.
- Job Migration-During failure of any task, it can be migrated to another machine. This technique can be implemented by using HAProxy.
- SGuard- It is less disruptive to normal stream processing and makes more resources available. SGuard is based on rollback recovery and can be implemented in HADOOP, Amazon EC2.
- Retry-It is the simplest task level technique that retries the failed task on the same cloud resource.
- Task Resubmission-It is the most widely used fault tolerance technique in current scientific workflow systems. Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at runtime.
- User defined exception handling-In this user specifies the particular treatment of a task failure for workflows.
- Rescue workflow-This technique allows the workflow to continue even if the task fails until it becomes impossible to move forward without catering the failed task [15].

2.3.2 Proactive Fault Tolerance

The rate of failures in larger systems is bound to increase, but proactive FT supports larger systems [14]. It is used to avoid failures, errors and faults by predicting them. Therefore, it replaces the suspected component by other currently working component that offers the same function. . Some of the techniques which are based on these policies are Preemptive migration, Software Rejuvenation etc.

- Software Rejuvenation-It is a technique that designs the system for periodic reboots. It restarts the system with clean state.
- Proactive Fault Tolerance using Self- Healing- When multiple instances of an application are running on multiple virtual machines, it automatically handles failure of application instances.
- Proactive Fault Tolerance using Preemptive Migration- Preemptive Migration relies on a feedback-loop control mechanism where application is constantly monitored and analyzed [15].

2.4 Challenges of Implementing Fault Tolerance in Cloud Computing

Providing fault tolerance requires careful consideration and analysis because of their complexity, inter-dependability and the following reasons.

- There is a need to implement autonomic fault tolerance technique for multiple instances of an application running on several virtual machines [16].

- The customer cannot easily extract their data and programs from one site to run on another site. The solution is to standardize the API's, so that SaaS developer could deploy services and data across multiple providers [29].

- Organizations worry about the whether the utility computing services will be adequately available or not. This provides the opportunity for multiple vendors to provide Cloud computing services [29].So, availability of the cloud services can be increased by using fault tolerance technique.
- Different technologies from competing vendors of cloud infrastructure need to be integrated for establishing a reliable system [17].

- The new approach needs to be developed that integrate these fault tolerance techniques with existing workflow scheduling algorithms [18].
- A benchmark based method can be developed in cloud environment for evaluating the performances of fault tolerance component in comparison with similar ones [19].
- To ensure high reliability and availability multiple clouds computing providers with independent software stacks should be used [20] [21].
- Autonomic fault tolerance must react to synchronization among various clouds [17].

2.5 VM Migration

It refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the virtual machine are transferred from the original host machine to the destination. Migration of a virtual machine is simply moving the VM running on a physical machine (let's call it source node) to another physical machine (let's call it target node). The trick is to do this, while the VM is running on the source node, and without disrupting any active network connections even after the VM is moved to the target node. It is considered "live", since the original VM is running, while the migration is in progress. In virtualization community, live migration of virtual machines is pretty much considered a "default" mature feature [22].

Virtual machine (VM) technology has recently emerged as an essential building block for data centers and cluster systems, mainly due to its capabilities of isolating, consolidating and migrating workload Altogether, these features allow a data center to serve multiple users in a secure, flexible and efficient way. Consequently, these virtualized infrastructures are consider a key component to drive the emerging Cloud Computing paradigm. Migration of virtual machines seeks to improve manageability, performance and fault tolerance of systems. More specifically, the reasons that justify VM migration in a production system include: the need to balance system load, which can be accomplished by migrating VMs [23].

2.5.1 Achieving live migration

To move a VM from the source node to the target node, we need to consider moving its CPU state, memory content, storage content, and network connections.

- Migrating CPU state
- Migrating memory content - The VM on the source node is still running and making modifications to the memory state. The idea is to do iterative copying of the memory contents, and send only the "delta" changes to the target node. There is a point, when only a small "delta" memory that needs to be copied. At this stage, the VM on the source node is paused, the delta memory is copied, and VM is resumed on the target node.
- Migrating storage content – It is similar to memory, but will require a lot more time, and the migration may take on the order of minutes.
- Migrating network connections- It is simple, if you assume that all of the nodes are in the same IP subnet. When the VM is migrated to the target node, the VM simply has to send an ARP broadcast saying that the IP address has moved to a new physical (MAC address) location. Since this happens at the connection between Layer 2 and Layer 3 of network stack, transport layer is transparent to this change, and TCP connections survive the migration. As a result, applications see no disruption in network connections. Clearly, this approach doesn't work, if the VMs have to cross subnets [22].

2.5.2 Benefits of VM migration

One of the primary use-cases for live migration is for resource management in cloud computing. For example, cloud computing providers like Amazon EC2 have thousands of VMs running in their data centers. To save energy, cost and for load balancing, they can move VMs using live migration, without disrupting their customer applications running in the VMs [22].

2.6 Dynamic Consolidation of Virtual Machines in Cloud Data Centers

Cloud computing model introduced the creation of large - scale virtualized datacenters. These datacenters produce a great amount of operational cost and environmental impact. Cloud providers enhance the resource usage and reduce energy consumption by dynamic consolidation of virtual machines (VM) using live migration and also by converting idle nodes to sleep mode. But vigorous consolidation leads to the degradation of performance and thus may not be able to provide for the SLA with the customers of the service provider. The VM placement must be optimized in continuity in an online manner to handle variability of workload. Thus, competitive analysis was conducted to understand the implications of the online nature of the problem. It also proved the competitive ratios of optimal online deterministic algorithms for single VM migration and dynamic VM consolidation by taking historical data .This proposed algorithm reduced the energy consumption and provided obedience to the service level agreements. Thus the tradeoff between the carbon footprint, operational costs and the performance and the effect of the implementation of various workload model algorithms on these parameters is studied in the paper. The experiments conducted on large scale setup of thousands of Planet Lab VMs have provided the much needed insight into these performance attributes.

2.6.1 Adaptive Heuristics for Dynamic VM Consolidation

Several heuristics are used for dynamic consolidation of VMs based on an analysis of historical data of the resource usage by VMs [28]. Dynamic VM consolidation is divided into four parts.

- Determining when a host is considered as being overloaded requiring migration of one or more VMs from this host
- Determining when a host is considered as being under loaded leading to a decision to migrate all VMs from this host and switch the host to the sleep mode
- Selection of VMs that should be migrated from an overloaded host
- Finding a new placement of the VMs selected for migration from the overloaded and under loaded hosts.

2.6.1.1 Host Overloading Detection

This method is based on the idea of setting upper and lower utilization thresholds for hosts and keeping the total utilization of the CPU by all the VMs 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 in order to prevent a potential SLA violation. However, fixed values of utilization 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, novel techniques are used for the auto-adjustment of the utilization thresholds based on a statistical analysis of historical data collected during the lifetime of VMs.

2.6.1.2 Local Regression

The main idea of the method of local regression is fitting simple models to localized subsets of data to build up a curve that approximates the original data. The observation (x_i,y_i) are assigned neighborhood weights using the tricube weight function shown as:

$$T(u) = \begin{cases} (1 - |u|^3)^3 & \text{if } |u| < 1, \\ 0 & \text{otherwise,} \end{cases}$$

Let $\Delta_i(x) = |x_i - x|$ be the distance from x to xi, and let $\Delta_i(x)$ be these distances ordered from smallest to largest. Then the neighborhood weight for the observation $(x_i, y_i)$is defined by the function $w_i(x)$

$w_i(x)= T\{(\Delta_i)(x)/\Delta_q(x)\}$

for x_i such that $\Delta_i(x) < \Delta_q(x)$, where q is the number of observations in the subset of data localized around x. The size of the subset is defined by a parameter of the method called the bandwidth. For example, if the degree of the polynomial fitted by the method is 1, then the parametric family of functions is y = a + bx. The line is fitted to the data using the weighted least squares method with weight $w_i(x)$ at(x_i, y_i). The values of a and b are found by minimizing the function

$$\sum_{i=1}^{n} w_i(x)(y_i - a - bx_i)^2.$$

,

2.6.1.3 VM Selection policies

Once it is decided that a host is overloaded, the next step is to select particular VMs to migrate from the host. After the selection of a VM to migrate, the host is checked again for being overloaded. If it is still considered being overloaded, the VM selection policy is applied iteratively. This is repeated until the host is considered as being not overloaded.

Three policies are:

1. **The Minimum Migration Time Policy**: It migrates a VM v that requires the minimum time to complete a migration relatively to the other VMs allocated to the host. The migration time is estimated as the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host.
2. **The Random Choice Policy**: It selects a VM to be migrated according to a uniformly distributed discrete random variable, whose values index a set of VMs allocated to a host.
3. **The Maximum Correlation Policy**: The idea is that the higher the correlation between the resource usage by applications running on an oversubscribed server, the higher the probability of the server overloading. So, VMs to be migrated which have the highest correlation of the CPU utilization with other VMs.To estimate the correlation between CPU utilizations by VMs, the multiple correlation coefficients is applied. [28]

Chapter 3

Experimental Set-Up

3.1 CloudSim

CloudSim: A new generalized and extensible simulation framework that enables seamless modelling, simulation, and experimentation of emerging Cloud computing infrastructures and management services. The simulation framework has the following novel features:

- Support for modelling and instantiation of large scale Cloud computing infrastructure, including data centers on a single physical computing node and java virtual machine.
- A self-contained platform for modelling datacenters, service brokers, scheduling, and allocations policies.
- Availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node.
- Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services [24].

3.1.1 CloudSim Architecture

Figure 3.1 shows the multi-layered design of the CloudSim software framework and its architectural components. Initial releases of CloudSim used SimJava as discrete event simulation engine [25] that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, virtual machines), communication between components, and management of the simulation clock. However in the current release, SimJava layer has been removed in order to allow some advanced operations that are not supported by it.

The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for virtual machines (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, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement their 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 SaaS provider's defined QoS levels. This layer also exposes functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer 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 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. As Cloud computing is still an emerging paradigm for distributed computing, there is a lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. Hence, in the near future there would be a number of research efforts both in academia and industry towards defining core algorithms, policies, and application benchmarking based on execution contexts. By extending the basic functionalities already exposed with CloudSim, researchers will be able to perform tests based on specific scenarios and configurations, thereby allowing the development of best practices in all the critical aspects related to Cloud Computing [26]

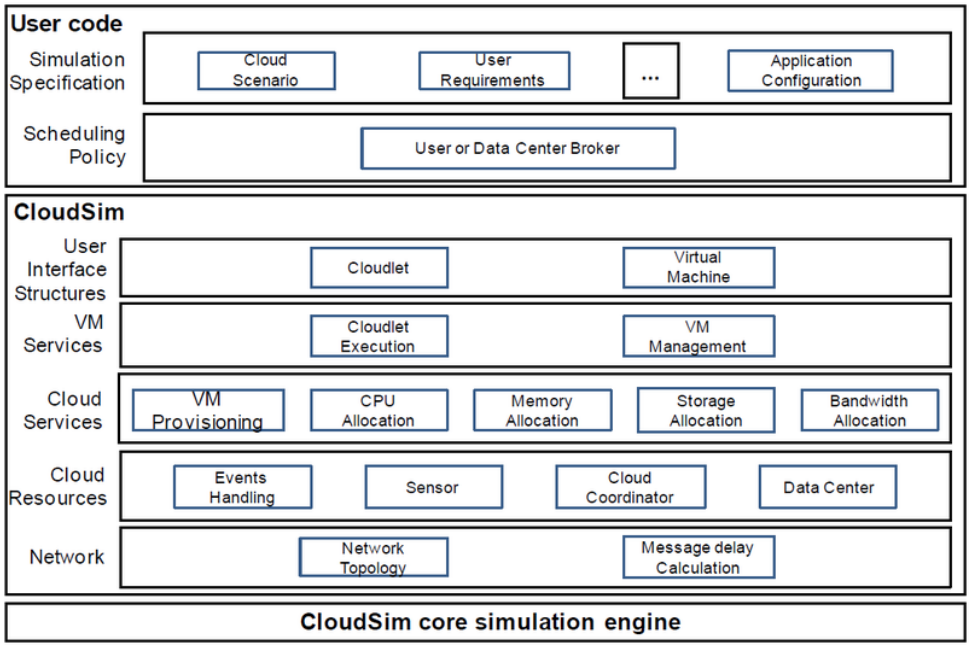

Figure 3.1 - Layered CloudSim architecture [26].

3.2 GridSim

The GridSim simulation framework enables users to

- model and simulate parallel and distributed computing systems
- determine effective and efficient resource allocation through simulation
- explore the significance of local economy and global positioning of resources

Figure 3.2 below gives a generic overview of a grid computing environment demonstrating the connections between resource brokers, information services, and resource distribution nodes.

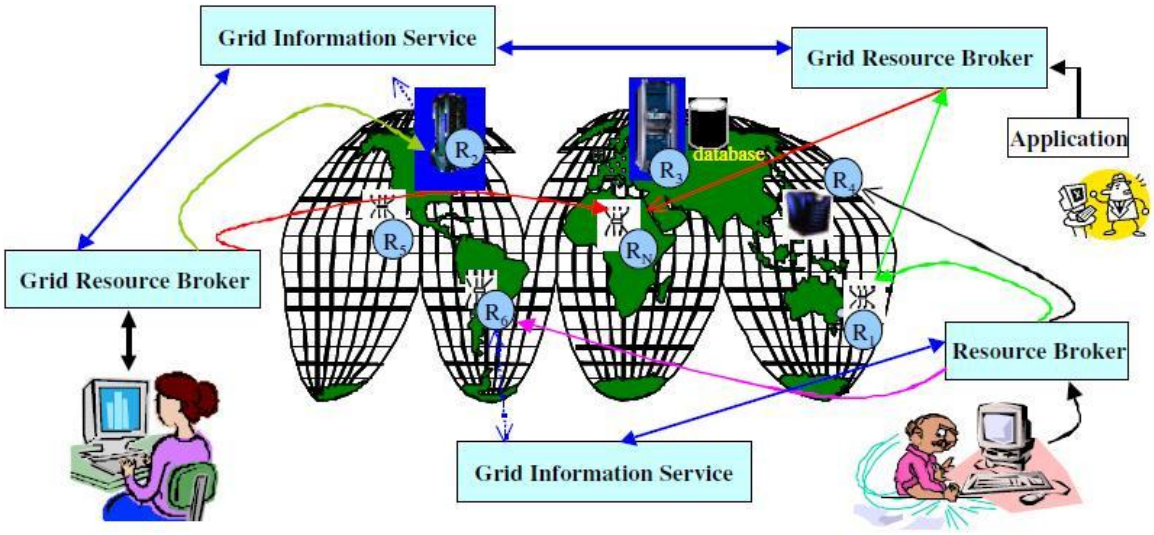

Figure 3.2 -Generic Overview of Grid Computing [27]

3.2.1 GridSim Features

GridSim features include:

- incorporating Grid resource failures during runtime
- supporting advance reservation of a grid system
- reading workload traces from supercomputers to simulate a realistic grid environment
- supporting data grid simulations
- utilizing a network topology to link together resources and other entities
- simulating background network traffic based on a probabilistic distribution to simulate a congested public network
- simulating an experiment with multiple virtual organizations

3.2.2 GridSim Architecture

The GridSim architecture shown below in figure 4.3 is built as a set of layered components. The base foundation is SimJava that is used for communication between the GridSim components. Forming the second layer are the Core Elements that are used to model Grid resources. The third and fourth layers model the services specific to computational and data grids respectively. Finally, the fifth and sixth layers consist of components to assist users in implementing schedulers, testing algorithms, defining scenarios, and creating configurations to validate algorithms [27].

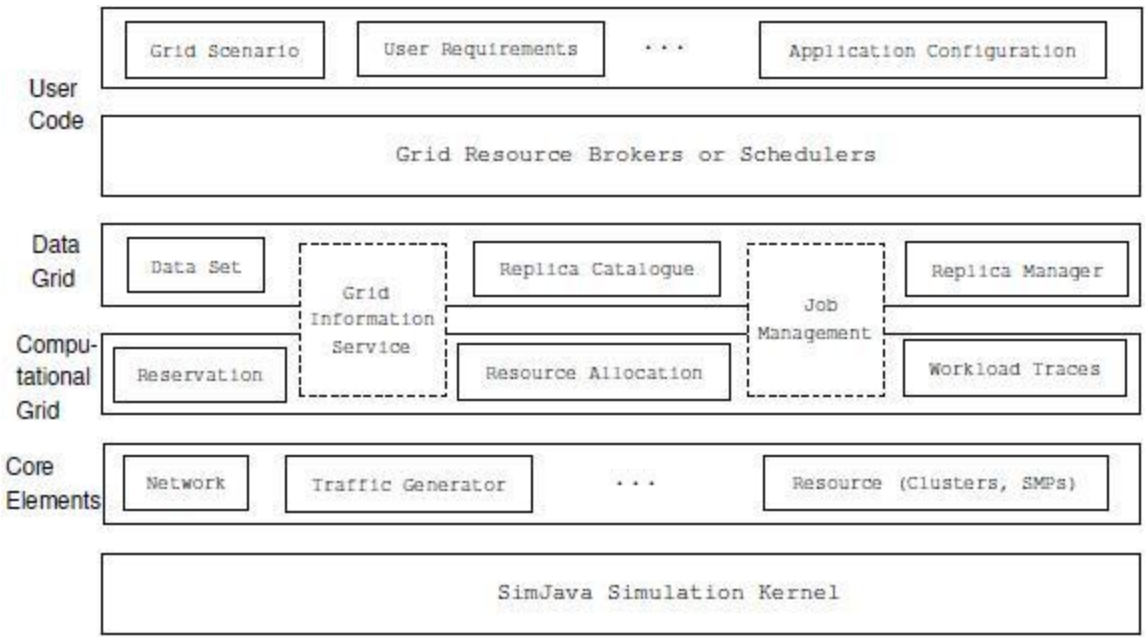

Figure 3.3 - GridSim Architecture [27]

3.3 The System Model

The targeted system is an IaaS environment, represented by a large-scale datacenter consisting of N heterogeneous physical nodes. Each node i is characterized by the CPU performance defined in Millions Instructions Per Second (MIPS), amount of RAM and network bandwidth. The servers do not have local disks, the storage is provided as a Network Attached Storage (NAS) to enable live migration of VMs. The type of the environment implies no knowledge of application workloads and time for which VMs are provisioned. Multiple independent users submit requests for provisioning of M heterogeneous VMs characterized by requirements to processing power defined in MIPS, amount of RAM and network bandwidth. The fact that the VMs are managed by independent users implies that the resulting workload created due to combining multiple VMs on a single physical node is mixed. The mixed workload is formed by various types of applications, such as HPC and web-applications, which utilize the resources simultaneously. The users establish SLAs with the resource provider to formalize the QoS delivered. The provider pays a penalty to the users in cases of SLA violations.

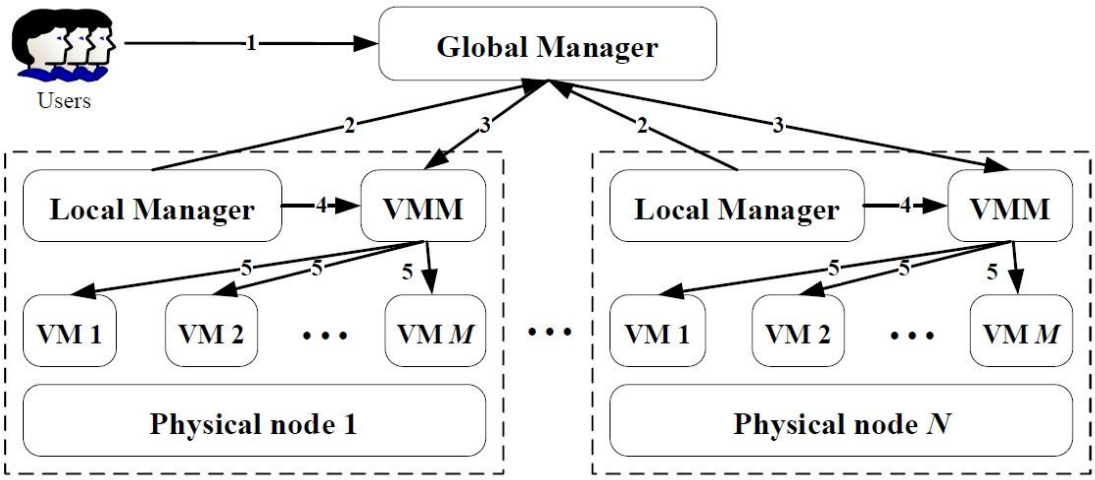

Figure 3.4- The system model [28]

The software layer of the system is tiered comprising local and global managers (Figure 3.4). The local managers reside on each node as a module of the VMM. Their objective is the continuous monitoring of the node's CPU utilization, resizing the VMs according to their resource needs, and deciding when and which VMs should to be migrated from the node. The global manager resides on the master node and collects information from the local managers to maintain the overall view of the utilization of resources. The global manager issues commands for the optimization of the VM placement. VMMs perform actual resizing and migration of VMs as well as changes in power modes of the nodes [28].

Chapter 4

Problem Statement

4.1 Gap Analysis

Based on understanding the utilization pattern of hosts, if there is overutilization of datacenters resources, the techniques implemented earlier tries to build a proactive response based on data points (data set of utilization metrics) by using localized subset of dataset, by which it is able to ignore the requirement for the specification of a function to fit a model, and only smoothing parameter value and degree of local polynomial is required.

However, this technique has certain disadvantages, and therefore need some improvements. The local regression technique used in identifying overload or over subscription of VMs, it only works if there is fairly large, densely sampled dataset for it to produce good model as it does local fitting, and it is difficult to represent by some mathematical formula, which makes it difficult to transfer results properly to other sub-systems of cloud to take decision in allocation of resources, and lastly this technique is prone to outlies for each local dataset. Each local data set creates its own threshold values applicable to local dataset. Sometimes there is a need for global threshold also to take decision because of factors like computational cost, low comprehensibility and storage requirements.

There is also a need to maintain idealize thresholds or the SLA thresholds, which needs to be maintained no matter what is the local threshold of each sub-data set, therefore there is an urgent need to take this into account, even if it comes out to be a case of static threshold.

4.2 Need and significance of research

The provision of host to the virtual machine in a datacenter needs a careful thought process. First, it must address how the host must be reserved, and then if those reservations come under over-subscription, over-utilization and performance degradation then how VM allocation policy must change to address these issues to that datacenters to take right decisions to re-allocate or migrate the VM machines. One way to mitigate such issues is to

design algorithms that can detect over utilization of resources and predict any adversity in datacenter. Once this is done, we can build a fall back VM allocation policy or in simple words fault tolerance mechanism which may be reactive or proactive in nature. Therefore there is an urgent need to build VM allocation policy that works as fault tolerance algorithm to migrate and consolidate the virtual machines dynamically and efficiently based on non-linear nature of distribution of workflow jobs in tasks.

Chapter – 5

Proposed Statement

5.1 Solution to the Problem

The solution of the problem is based on understanding the utilization pattern of the host. To overcome the issues, kernel regression, which is specific to the issue, is applied, in which distance measures like Euclidean Distance can be used to form model for taking VM allocation decisions. Since, dynamic workload exists in which CPU usage varies with time , in this case, if CPU usage exceeds the available capacity of CPU, it is considered that there exists deviation from idealized threshold. Thus, there is a need of re-allocation or migration of VMs or simply a fall back mechanism that runs datacenter without performance degradation based on work variability.

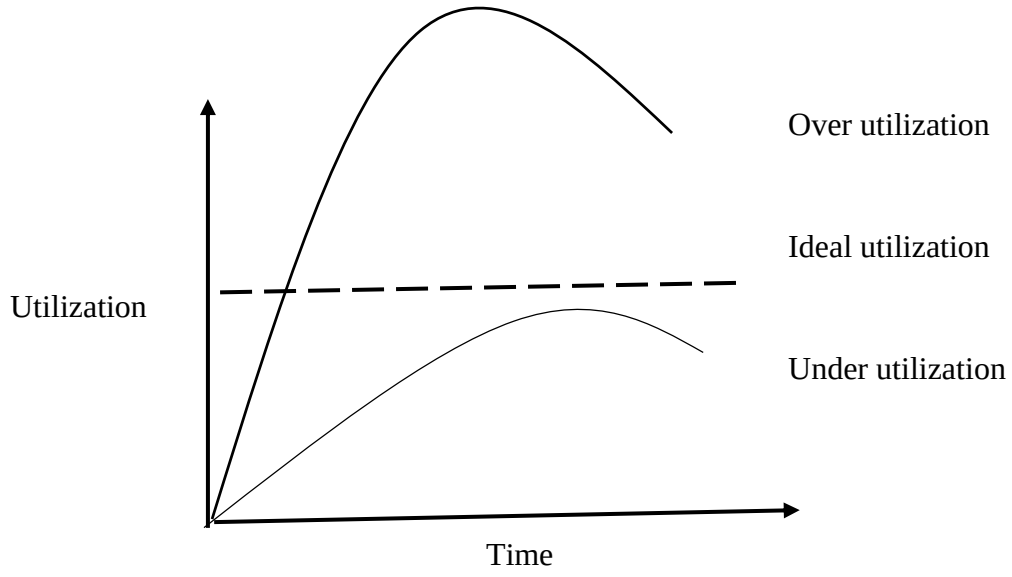

Figure 5.1- Utilization vs. Time Graph Based on Work Availability

Once, the deviation from ideal utilization threshold is detected, there is a need to form VM selection policy to complete the fault tolerance mechanism which provides smooth running of datacenters.

Improvement on one of the following can be made –

- Random Choice Policy
- Maximum Correlation Policy
- Minimum Migration Time

In this research work, improvement in Maximum Correlation Policy is improvised as correlation cannot explain the cause of two random variable's association, that is, how these variables are positively or negatively co-related.

Therefore, for a VM policy to find a 'v' virtual machine, that satisfies (equal number 38) may be a good idea , as correlation is normally bivariant in nature, as they compare two numbers only at a time from two datasets. There are almost multiple relationship sand effects due to multiple factors like payload, time zone, etc. Therefore, for improving this algorithm of VM selection a new solution is suggested which not only considers correlation between CPU utilization and machine to be allocated, but also consider other correlations between other factors and the machine to be allocated to form a score matrix, which helps to finally take decision in choosing the best machine.

5.2 Performance Evaluation

The selection of performance parameters should be done in such a manner that it is able to find the efficiency of the algorithm in terms of time and must be able to reduce failures and help in running of datacenters smoothly.

The descriptive statistical parameters include

- Mean
- Standard Deviation
- Variance

Since the workload is random in nature, there is high variability in terms of processing the workflow and expected number of possible migrations due to overload. Therefore, the best way to understand the distribution of all the process of fault tolerance mechanism (VM allocation and VM selection) is to find mean of each parameters, and if there is any deviation from the mean, there is a need to calculate how much it deviates from the mean value which is done by calculating the variance and standard deviation.

These three parameters gives an overall picture of the nature of the fault tolerance mechanism for each step it performs. For example, it helps us to understand how much average time it takes for the algorithm to give response for reallocating or migrating the workload to a new VM and if there is some variation in doing this process we also get how much is the variation by calculating standard deviation and variance.

5.3 Proposed Approach

5.3.1 Multiple-maximum Correlation Policy

Our proposed approach is an extension of the multiple correlation analysis used in base paper, it is a method of describing complex sequential relationship between measurements. Conceptually, the variables (measurements) used in the model are assumed to be all the measurements that matters. These are assigned to different levels in a sequence or cascade of influence, where the earlier levels affect the subsequent ones, but the reverse does not happen. In this sequence, all measurements in prior levels affect all subsequent levels, and the scale of the influence is described by the path coefficient, and partial correlation between measurements in the same level assumes that there are as yet unexplained common preceding influences. Mathematically, it is called path analysis, which consists of a repeated sequence of multiple correlation calculations from a correlation matrix, following the cascade of influences. This is carried out one level at a time, using all the measurements in preceding levels as independent variables, and the Standardized Partial Regression Coefficients (Path Coefficients) represents the size of each influence. This is followed by calculating Partial Correlation Coefficients between all the variables in the same level, corrected for all preceding variables as well. These Partial Correlation Coefficients represents common influences that have not as yet been explained by the model.

For example table 5.1 shows correlation matrix that shows the relationship between the CPU usage and VMs.

Table 5.1 correlation matrix

	VM1	VM2	VM3	VM4	VM5	VM6	VM7
VM1	1	0.5	0.2	0.2	0.3	0.6	0.4
VM2	0.5	1	0.1	0.1	0.3	0.7	0.5
VM3	0.2	0.1	1	0.4	0.5	0.1	0.5
VM4	0.2	0.1	0.4	1	0.6	0.1	0.4
VM5	0.3	0.3	0.5	0.6	1	0.3	0.7
VM6	0.6	0.7	0.1	0.1	0.3	1	0.8
VM7	0.4	0.5	0.5	0.4	0.7	0.8	1

Let VM7 represent one of the VMs that is currently considered for being avoided for migration. but , it might happen that .Given that CPU utilization of VM1,VM2 may be correlated with those of VM3 and VM4 as also have similar pattern of CPU utilization, and VM5 and VM6 also may have similar ,CPU utilization pattern to VM1 and VM2, we need a multiple correlation analysis to correct for all the inter-correlations, and identify the direct influence (Partial Correlation Coefficient) each member has on the selection VM7 as candidate which will have low performance and further degradation due over utilization . The VM7, is the machine is which is expected to have maximum load of the work or may have overload, and thus might degrade in performance as time comes, there for, a fallback policy is required if the VM7 fails or is overloaded with workload, hence, we run our fault tolerances to overcome such issue, here, this machine will be avoided in future re-allocation of machines or migrations of machines and the one which has the best performance will be considered for VM migration.

Chapter-6

Experimental Results

6.1 Implementation and Working of the proposed approach

Following experiments were conducted to identify that how good is the VM migration policy. Experimental results shows in the form of tabular form and graphical form.

Table-6.1 Evaluation of Fault Tolerance Mechanism

Experiment Number	Overloading Detection Algorithm	VM Migration Algorithm	Safety Parameter	No Of VM/Host	Workload Type
1	Local Regression	Max Corr	0.5	50	Random
2	Local Regression	Max Corr	0.6	50	Random
3	Local Regression	Max Corr	0.7	50	Random
4	`Local Regression	Max Corr	0.8	50	Random
5	Local Regression	Max Corr	0.9	50	Random
6	Local Regression	Max Corr	1	50	Random
7	Local Regression	Max Corr	1.2	50	Random
8	Local Regress	Proposed	0.5	50	Random
9	Local Regression	Proposed	0.6	50	Random
10	Local Regression	Proposed	0.6	50	Random
11	Local Regression	Proposed	0.7	50	Random
12	Local regression	Proposed	0.8	50	Random

13	Local Regression	Proposed	0.9	50	Random
14	Local Regression	Proposed	1	50	Random
15	Local Regression	Proposed	1.2	50	Random

Table 6.1 shows that overload detection algorithm, with VM migration algorithm, safety parameter, No. of VM's, Safety parameter, workload type. Local regression algorithm is used with maximum correlation policy and with proposed multiple maximum correlation policy for 50 VMs .The VMs are allocated according to the resource requirements defined by the VM types.

Table - 6.2 Results of Experiments

Exeriment Number	Number of VM migrations	VM selection mean	VM selection stDev	host selection stDev	VM reallocation mean	VM reallocation stDev	Safety Parameter
1	257	0.00137	0.007	0.00531	0.00022	0.00182	0.5
2	361	0.00283	0.009	0.00561	0.00016	0.0016	0.6
3	817	0.00766	0.014	0.00535	0.00104	0.00392	0.7
4	1457	0.0103	0.015	0.00552	0.00141	0.00486	0.8
5	1735	0.01228	0.012	0.00512	0.00174	0.00491	0.9
6	2430	0.0135	0.015	0.00606	0.00394	0.00683	1
7	3081	0.01326	0.02	0.00589	0.00658	0.00846	1.2
8	273	0.00109	0.006	0.00499	0.0002	0.0017	0.5
9	377	0.00201	0.009	0.00559	0.000155	0.0012	0.6
10	817	0.00166	0.013	0.005	0.001	0.00301	0.7
11	1457	0.0111	0.014	0.005	0.00114	0.00409	0.8

12	1735	0.01122	0.011	0.005	0.00155	0.003999	0.9
13	2430	0.01335	0.014	0.00588	0.00355	0.00513	1
14	3081	0.01325	0.02	0.005	0.006	0.005999	1.2

Table 6.2 shows the 14 experiments. Experimental results shows that it reduces the mean time, standard deviation mean, VM standard mean. VMs utilize less resources according to the workload data, creating opportunities for dynamic consolidation but if there is overloading or adversity it again forms another type of challenge in terms of defining the VM reallocation or migration policy after the overloading and adversity has been detected, which is the core part of any fault tolerance mechanism in cloud computing.

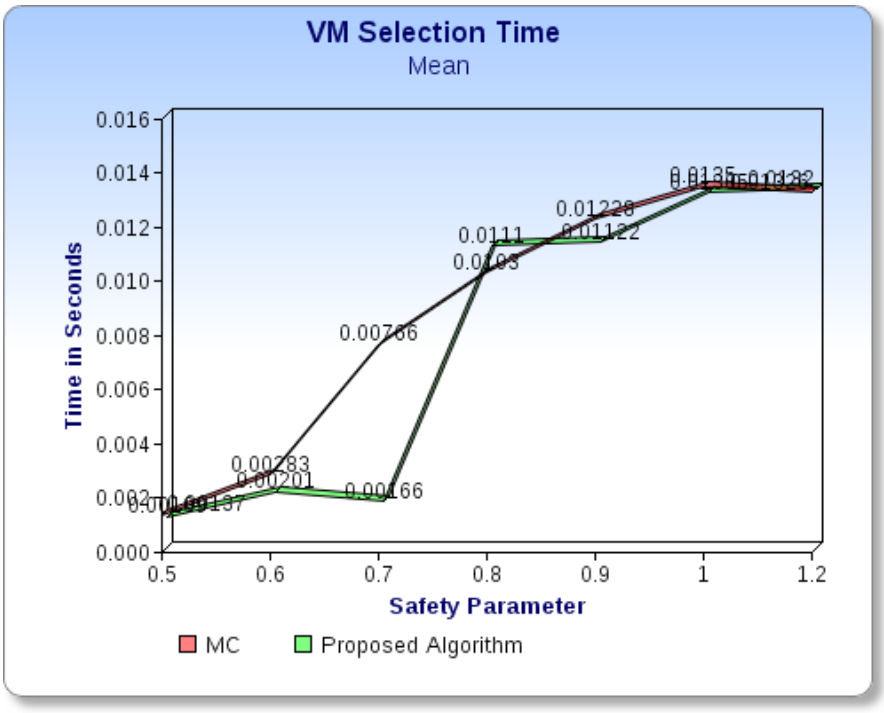

Figure 6.1- Graph of VM Selection Time Mean

It can be seen from the above graph in Figure 6.1, that the mean selection time for the Fault tolerance algorithm for the proposed algorithm is coming below the value of the Maximum Correlation (MC) algorithm in most of the cases but at the time when the safety parameter value is around 0.8 , the value goes above the MC value ,and then finally it has similar

magnitude value when it comes to safety parameter value of 1.2.From the above line curve we can also infer that the proposed algorithm overall VM selection policy is performing better as it is taking less average time in selecting the new VM placement when utilization varies due to safety parameter.

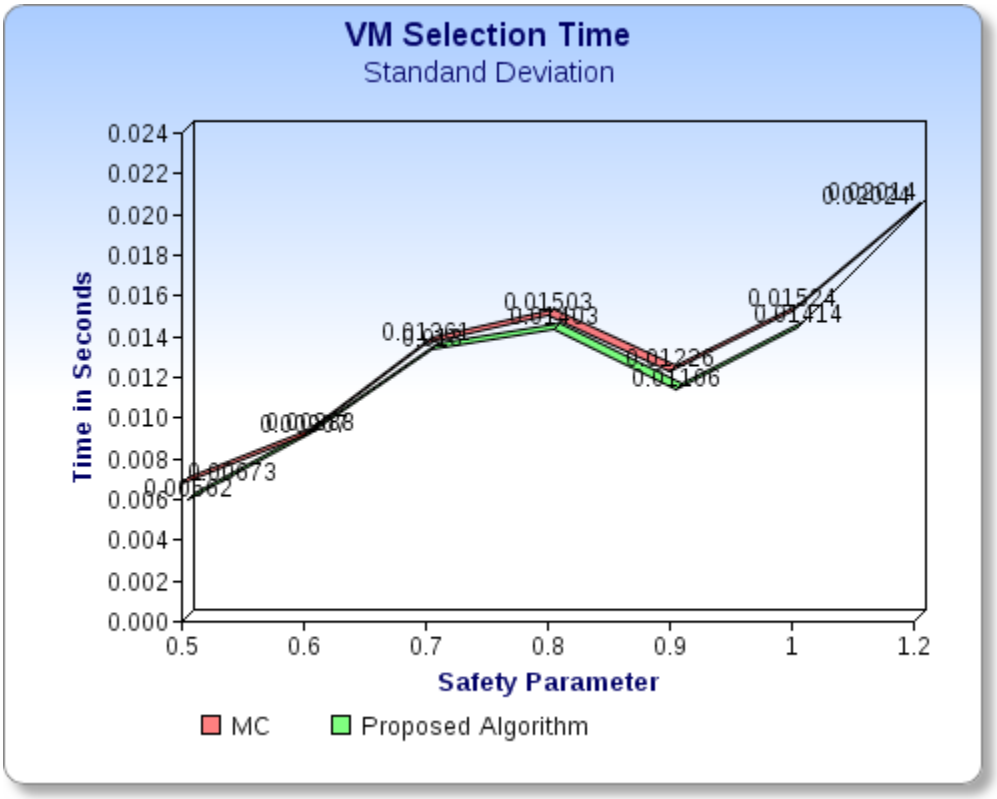

Figure 6.2- Graph of VM Selection Time Standard Deviation

From the above graph Figure 6.2 also we can apparently infer that proposed algorithm shown in green line is performing on the most of the levels of utilization, the curve showing lower deviation from the average time taken in VM selection, this may be attribute the fact the overloading detection algorithm (Local Regression) is same while initial selection of the VMs, but it is standard deviation value falls below as average value is also.

Chapter-7

Conclusion and Future Scope

This chapter discusses the conclusions of work presented in this thesis. The chapter ends with a discussion of the future direction which thesis work can take.

7.1 Conclusion

In this thesis VM migration policy is proposed and implemented.VM migration policy uses the multiple maximum correlation approach to migrate the Virtual Machine to ideal one. Overload detection is done using local regression and VM allocation is implemented using multiple maximum correlation approach. The experimental results provide us with the descriptive statistical parameters including mean, standard deviation and variance. These parameters are based on migration time, number of host handled, number of host overload detection, number of cloudlets etc.

7.2 Thesis Contributions

- Proposed a VM migration policy.
- Design the approach for overloaded detection.
- Implementation of VM migration policy in cloud environment.

7.3 Future Research

- This work shows implementation of the reactive fault tolerance approach. It can be used for proactive fault tolerance
- It can be used to handle multiple tasks.
- It can further be used to merge with scheduling algorithms.

References

[1] Bohm, M.,Leimeister.S.,et al., ”Cloud Computing and Computing Evolution,(TUM)”.

[2] Whyman,B.,”Cloud Computing industry trends FISMA”.

[3] Cloud Computing available at http://www.siteground.com/tutorials/cloud/cloud_computing.htm

[4] Gao,J.,Bai,X.,et.al.,” Cloud Testing- Issues, Challenges, Needs and Practice”.

[5] Book on Computing available at http://www.south.cattelecom.com/Technologies/CloudComputing/0071626948_chap01.pdf

[6] Kaushal, V.,Bala,A.,” Autonomic Fault Tolerance Using HAProxy in Cloud Environment”.

[7] Boniface, M., Nasser, B., Papay, J., Phillips, S., Servin, A., Yang, X., et al.,”Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds”.

[8] Murphy, M., Abraham, L., Fenn, M., & Goasguen, S.,” Autonomic Clouds on the Grid. Journal of Grid Computing”.

[9] Cloud Deployment Models available at http://www.cloudtweaks.com/2012/07/the-4-primary-cloud-deployment-models/

[10] Beloglazov,A.,and Buyya,R.” OpenStack Neat: A Framework for Dynamic Consolidation of Virtual Machines in OpenStack Clouds – A Blueprint”.

[11] Article on Fault Tolerance Computing at http://www.cs.ucla.edu/~rennels/article98.pdf

[12] Abd-El-Barr,M.,” Design and Analysis of Reliable and Fault-Tolerant Computer Systems”,

[13] Xie, Z., Sun,H., and Saluja,K., “A Survey of Software Fault Tolerance Techniques”.

[14] Nagarajan,A.B., Mueller,F., Engelmann, C.,and Scott,S.L.,” Proactive fault tolerance for HPC with Xen virtualization”.

[15] Bala,A.,Chana,I.,”Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing “.

[16] Chen, G.,Jin,H., Zou,D., et al., “SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment”.

[17] Abbadi, I.M.,“Self-Managed Services Conceptual Model in Trustworthy Clouds' Infrastructure”.

[18] Zhang,Y., Mandal,A., et al.,” Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids”.

[19] Hwang,S.,KesselmanC.,” Workflow: A Flexible Failure Handling Framework for the Grid”.

[20] Armbrust, M., Fox, A., Griffith, R., et al.,” Above the clouds: A Berkeley view of cloud computing”.

[21] Zhao,W., Melliar-Smith,P.M., and L. E. Moser,” Fault Tolerance Middleware for Cloud Computing”.

[22] Understanding Live Migration at http://ppadala.net/blog/2010/06/understanding-live-migration-of-virtual-machines/

[23] Voorsluys, W., Broberg,J., et al., “Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation”

[24] Calheiros,R.N., Ranjan,R., et al.,“CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services”.

[25] Howell,F., and Mcnab,R.,” SimJava: A discrete event simulation library for java”.

[26] Calheiros,R.N., Ranjan,R., et al.,“CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithm”.

[27] Cloud Simulation available at http://cloud-simulation-frameworks.wikispaces.asu.edu/

[28] Beloglazov,A., and Buyya,R.,“Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers”.

[29] Kumar,P.,” Abstract Model of Fault Tolerance Algorithm in Cloud Computing Communication Networks”.

[30] Malik, S., Huet,F.,” Adaptive Fault Tolerance in Real Time Cloud - Computing”.