Dependability Differentiation in Cloud Services

Ameen Chilwan

Master of Telematics - Communication Networks and Networked Services (2 year)
Submission date: July 2011
Supervisor: Poul Einar Heegaard, ITEM
Co-supervisor: Astrid Undheim, Telenor

Norwegian University of Science and Technology
Department of Telematics

NTNU

Norges Teknisk-Naturvitenskapelige Universitet (NTNU)

Dependability Differentiation in Cloud Services

TTM4905: Networks and Services
Master's Thesis

Ameen Chilwan

7/4/2011

Supervised By: Astrid Undheim

Professor: Poul E. Heegaard

Thesis Description

Cloud computing is an evolving computing paradigm in which computing is provided to consumers upon request as a service and is charged on pay-as-you-go basis. The virtually infinite computing resources on the cloud provider side and the dramatic decrease of investment requirement on the consumer side have made large companies to consider outsourcing their IT services to cloud providers.

But, large companies have very strict dependability requirements if they have to outsource their internal IT functions to clouds. Their requirements differ largely from a domestic cloud user. The cloud providers are, thus, supposed to provide different levels of dependability to different types of users depending upon the SLAs between them.

One of the dependability attributes for cloud services is service availability which is the availability perceived by a cloud user. It is an umbrella attribute which gathers cloud availability, network availability, cloud performance, network performance and cloud security under it. It is also one of the most important dependability requirements of most of the large IT functions.

In the project work, autumn 2010, analytical models for differentiating cloud availability by replicating Virtual Machines (VMs) were developed, using different replication schemes. This thesis work will build upon these models, refining them where necessary, and analyzing the achieved dependability differentiation. In addition to that, investigations to find other techniques for dependability differentiation both on the cloud and the network side (both availability and performance) will be made. Service availability will then be accounted for as a whole and comparisons will be made to find out that on which part of the cloud service provision (cloud or network) it is worth investing in terms of resources and finance to achieve better differentiation.

The following tasks will be carried out in thesis:

1. Investigate and identify possible techniques for dependability differentiation both in clouds and networks.
2. Identify example services with different dependability requirements:
    a. Single service with different dependability requirements for different user classes
    b. Different services with different dependability requirements
3. Examine the models from the project work with respect to simplifications and realism
4. Develop models for a selected set of scenarios with differentiated dependability in both clouds and networks for example service(s).
5. Simulate the scenarios and compare them.
6. Discuss which part of cloud service provision (cloud or network) is worth investing in, in terms of resources and finance, to achieve better dependability differentiation.

Assignment Given: **February, 2011**

Professor: **Poul E. Heegaard**

Supervisor: **Astrid Undheim**

Abstract

As cloud computing is becoming more mature and pervasive, almost all types of services are being deployed in clouds. This has also widened the spectrum of cloud users which encompasses from domestic users to large companies. One of the main concerns of large companies outsourcing their IT functions to clouds is the availability of their functions. On the other hand, availability requirements for domestic users are not very strict. This requires the cloud service providers to guarantee different dependability levels for different users and services. This thesis is based upon this requirement of dependability differentiation of cloud services depending upon the nature of services and target users.

In this thesis, different types of services are identified and grouped together both according to their deployment nature and their target users. Also a range of techniques for guaranteeing dependability in the cloud environment are identified and classified. In order to quantify dependability provided by different techniques, a cloud system is modeled. Two different levels of dependability differentiation are considered, namely; differentiation depending upon the state of standby replica and differentiation depending upon the spatial separation of active and standby replicas. These two levels are separately modeled by using Markov state diagrams and reliability block diagrams respectively. Due to the limitations imposed by Markov models, the former differentiation level is also studied by using a simulation.

Finally, numerical analysis is conducted and different techniques are compared. Also the best technique for each user and service class is identified depending upon the results obtained. The most crucial components for guaranteeing dependability in cloud environment are also identified. This will direct the future prospects of study and also provide an idea to cloud service providers about the cloud components that are worth investing in, for enhancing service availability.

Acknowledgement

First and the foremost, I thank **Allah the Almighty** for giving me health, strength and courage to finish my master’s thesis and eventually M.Sc. in Telematics at NTNU.

Secondly, I would cordially like to acknowledge the constant support of my supervisor, **Astrid Undheim.** Her dedication, determination and extended interest in my work throughout the year have proved to be instrumental in reaching my goals. She motivated me when I was disheartened and uplifted me when I was down. I owe her a great deal.

This thesis wouldn’t be possible without the valuable comments of my professor **Poul E. Heegaard**. I am grateful to him for always finding time for me from his busy schedule and giving me precious advices.

This acknowledgement would be incomplete if I don’t mention **Zeeshan**, my flat-mate and best friend, and thank him for all what he has done for me. He made my stay in Norway enjoyable and proved to be a moral strength to me.

Last but not the least, I would like to offer my heartily gratitude to my family, especially my mother, **Veena**, and my siblings; **Amna**, **Asma**, **Afrah, Aiman** and **Talaa** for their concern and blessings. Thank you all.

Table of Contents

List of Tables

List of Figures

List of Acronyms

CaaS – Communication-as-a-Service

CPU – Central Processing Unit

DaaS – Data-Storage-as-a-Service

DDoS – Distributed Denial of Service

DEMOS – Discrete Event Modeling On Simula

DMTF – Distributed Management Task Force

EDG – Electrical Diesel Generators

FT – Fault Tolerance

HA – High Availability

HDD – Hard Disk Drive

IaaS – Infrastructure-as-a-Service

ICT – Information and Communication Technology

LTE – Long Term Evolution

n.e.d – Negative Exponential Distribution

NIC – Network Interface Card

NIST – National Institute of Standards and Technology

NIX – Norway Internet eXchange

OS – Operating System

OVF – Open Virtualization Format

PaaS – Platform-as-a-Service

PDA – Personal Digital Assistant

PDU – Power Distribution Unit

PRS – Physical Resource Set

RPO – Recovery Point Objective

RTO – Recovery Time Objective

SaaS – Software-as-a-Service

SLA – Service Level Agreement

SNMP – Simple Network Management Protocol

STS – Static Transfer Switch

UPS – Uninterrupted Power Supply

VM – Virtual Machine

VMM – Virtual Machine Monitor

VRS – Virtual Resource Set

1 Introduction

1.1 Background

Cloud Computing is an evolving computing paradigm. It is projected that computing will be supplied to people as the fifth utility [1], in addition to water, gas, electricity and telephone. In its essence, cloud computing is an environment in which computing is done somewhere in the **cloud** and not necessarily on the physical machine on which it is needed. As cloud computing is becoming more mature and pervasive, almost all types of services are being deployed in clouds. This has also widened the spectrum of cloud users which encompasses from domestic users to large companies.

Many large companies are now considering outsourcing their consumer-centric services and even internal IT services to cloud providers, or deploying these in private, company internal clouds. A major concern in cloud computing, however, remains the availability of the service to them. Service availability, in this context, is referred to the user's experience of the availability of the service [2]. On the other hand, availability requirements for domestic users are not very strict. Only a basic level of availability of service is sufficient for a domestic cloud user.

In the cloud computing domain, the service unavailability is argued to be not only comprised of the unavailability of the network and the cloud, but also the compromised performance of the two and security breaches in the cloud. Because, in the user's experience, low performance of cloud or network, or Distributed Denial of Service (DDoS) security breach in the cloud, all lead to service unavailability. This generalization however remains quite idealistic. The dependability and performance attributes should be dealt with separately as proposed in [3]. Presently, popular could providers offer cloud availability as high as 99.95% [4] that is about 4.5 hours of downtime in a year, which is quite low for many services, e.g. switching, control systems etc. [5].

1.2 Thesis Motivation

In this thesis, the main concern is to quantify dependability guaranteed by different techniques already present in the market and compare them. Then, it is needed to comment that which technique best suits which service and user, depending upon their dependability requirements. The motivation behind this study is that different users have different dependability requirements and their willingness to pay for higher dependability also differs. Corporate users with high dependability requirement are willing to pay for higher dependability, while domestic users just need an acceptable level of dependability. Hence each of them needs to have a different Service level Agreement (SLA) with the cloud service provider. In the same course, it will be possible for the cloud service providers to market their products by introducing differentiated service classes for different groups of users.

So far, the SLAs of large cloud services, for example; Amazon EC2 [4], Google Apps [6] and Windows Azure [7] provide flat availability values in their SLAs. This trend can be changed by dependability differentiation and hence differentiated SLAs will be possible to be drawn for different groups of users for different availability levels at different costs. A step towards differentiated dependability is taken by Amazon as it charges users for providing them an extra standby replica running in different geographical location but the resulting availability is not quantified. This thesis will quantify availability for all such scenarios.

1.3 Thesis Scope

Availability, of both cloud and network, is chosen as the dependability attribute used for performing studies in this thesis. The driving force behind this choice is that most of the services provided in clouds currently have clear availability requirements while not very rigid reliability requirements. To understand this behavior of cloud services, it is instrumental to draw a line between these two dependability attributes.

Availability of a service is the percentage of time it is available in a certain span of time while reliability is the probability of providing uninterrupted service for certain period of time [5]. The choice of availability as the preferred metric in this project is due to the nature of the services offered in the cloud environment currently. The examples of main cloud services are computing, storage, email, development platform etc. [8], [9], [10]. It is obvious that these services need to be up as much of the time as possible but the continuity in their service provision is not that crucial, unlike the aircraft control systems [5] and tele-surgery systems which may tolerate long downtimes, but when they are up, they need to provide uninterrupted service. Availability is also chosen because; most of the cloud providers guarantee a certain level of availability in their SLAs [4], [6], [7] as this metric is publicly more understandable. Also, availability can easily be measured and calculated.

Also, out of five different potential causes of service interruption and degradation; cloud unavailability, network unavailability, degraded cloud performance, degraded network performance and breached cloud security [11] only cloud availability and network availability are considered in this thesis. The reason for this choice is to keep the study simple and manageable at this level.

1.4 Thesis Goals

This thesis is aimed to fulfill the following goals:-

1. Investigate and identify possible techniques for dependability differentiation both in clouds and networks.
2. Identify example services with different dependability requirements:
    a. Single service with different dependability requirements for different user classes
    b. Different services with different dependability requirements
3. Examine the models from the project work [12] with respect to simplifications and realism
4. Develop models for a selected set of scenarios with differentiated dependability in both clouds and networks for example system
5. Simulate the scenarios and compare them.
6. Discuss which part of cloud service provision is worth investing in, in terms of resources and finance, to achieve better dependability differentiation.

1.5 Thesis Contribution

In this thesis, a number of contributions are made to the field of service classification and dependability differentiation in cloud computing paradigm. To the best of our knowledge, the following contributions are made by us which haven't yet been reported:-

1. Identifying criteria for classification of cloud services, both with respect to user dependability requirements and nature of the services, and performing the classification
2. Investigating dependability differentiating techniques and arranging them in a classification tree by identifying their core characteristics
3. Proposing a complete model for studying dependability of cloud, although works for studying availability of different levels exist [13], this approach is, to the best of our knowledge, not exploited.
4. Developing analytical models for studying dependability differentiation techniques and analyzing them.
5. Simulating a cluster of cloud service for studying dependability differentiation techniques

1.6 Thesis Organization

After giving the background information about the work to be done in this thesis and outlining the goals, limitations and contributions of it in this chapter, the rest of the report is organized as follows.

In Chapter 2, a brief introduction of cloud computing is given. It is followed by classification of cloud services with respect to different criteria in the same chapter. Chapter 3 is also an investigation chapter in which some dependability concepts to be used in this thesis are explained. But, in addition to that, it also contains the classification of dependability differentiation techniques considering the state of the standby replica.

Chapter 4 gives a structural overview of the cloud system to be studied in this thesis. The main failures in cloud systems are also tabulated and explained in the same chapter. In Chapter 5 analytical models, both Markov models and reliability block diagrams, are drawn, analyzed, simplified and solved for dependability differentiation techniques on different levels. In Chapter 6 the dependability differentiation study is enhanced by modeling and validating a simulator for simulating the dependability differentiation techniques with respect to the state of the standby replicas.

Chapter 7 starts with enumeration of parameters found in literature for numerically analyzing the models and simulator developed. It goes on with the presentation of results and their respective discussions. The thesis is concluded and an account on way forward is given in Chapter 8.

2 Cloud Services

2.1 Cloud Computing

Building upon the characteristics enumerated in [14] and [15], cloud computing is defined as, "A paradigm for providing **on-demand access** to the **shared pool** of computing resources (e.g. servers, networks, storage, services etc.) **over network** that can be readily **provisioned, configured and released** with minimal human intervention and are abruptly **scaled** to the user requirements. The user is billed following the **pay-per-use** utility model."

As with other novel concepts, cloud computing is defined in a number of ways by having different basis of definition. In some cases the definition is based upon the technologies used to realize cloud computing [1], while in other cases the salient features of cloud computing are used to define it [16]. In some other cases, even the business model for provision of cloud computing and economic benefits obtained from it is made the basis of its definition [17]. The latter is because it is believed that cloud computing can fulfill the long-awaited dream of providing computing as a utility to the users [1].

An extensive survey of cloud computing definitions is done in [15]. In that work, almost all the different basis of defining cloud computing are considered and the authors have tried to come up with an integrative definition and a basic definition. The integrative definition encompasses the aspects given in all the definitions while the basic definition gives the essential aspects for a paradigm to be called cloud computing in light of the 20 definitions reviewed in the paper.

It is, however, noted that the approach followed by researchers at NIST (National Institute of Standards and Technology) of defining cloud computing in terms of its characteristics is not very much exploited, even in the definitions reviewed in [15]. But we believe that this is the best way of defining cloud computing. And hence the fore-stated definition follows this methodology. This is because it makes cloud computing definition independent of the underlying technologies and hence, among other benefits, makes the definition future-proof.

Let's say, for example, one of the essential characteristics of cloud computing is resource pooling [16]. Currently, virtualization of servers is used to fulfill this characteristic but if in future a better technology will be introduced for achieving the same, then we won't have to re-define cloud computing. This is not the case with technology-based definitions, because in such definitions virtualization is postulated as the basic requirement for a paradigm to be called cloud computing paradigm [15].

2.1.1 Characteristics of Cloud Computing Paradigm

In light of the above definition, five essential characteristics of cloud computing are identified. A very short account of these characteristics along with the technologies which are currently used to provide these characteristics follows. A detailed account of the characteristic-technology mapping can be found in [11].

2.1.1.1 On-Demand Self Service

One of the very basic motives behind the development of cloud computing is the fulfillment of the vision of providing computing to users as a utility, just like electricity and telephone [1]. Hence, this requires computing services to be available to users when they need them, that is, on-demand. It also requires that it would be as easy and automated for the user to get connected to these services as it is for other utility services, this is the notion of self-service.

For this purpose, a management system is needed which can assist user in receiving the required service and automatically connect and disconnect him from the cloud. Hence the management system will take care of the establishment and termination of connection to the service provider whenever required and according to the requirements of the user. These requirements may be a specific level of availability, capacity, processing power or anything else. In addition to that, computing services are also required to be billed just like other utilities. Hence, a metering and monitoring system also has to be in place for billing the users accordingly.

The requirements needed to fulfill the on-demand user access are readily available now. There are quite a number of cloud service providers active in the market and they have deployed different management systems on different levels of abstraction. These management systems are not only used in managing user requests and billing them, but also manage the virtual machines running in the datacenters and implement the additional mechanisms required for specific purposes, for instance live migration, communication between databases and respective application servers etc. Thus, management system makes the fundamental entity of this study because the mechanisms for differentiating dependability of different services are implemented as a part of it. More details about cloud management system will follow in Section 2.2.2.

2.1.1.2 Shared Pool of Resources

Another very fundamental feature of cloud computing is the concept of having infinite resources available for everybody at every instance of time [17]. It is achieved by having a “pool” of resources which is shared by services deployed in cloud and any service in need of resources just reserves some for itself. The size of the pool is made apparently infinite [17].

In order to provide this characteristic of cloud computing, very large datacenters are built. The extent of resources available in these datacenters is increased dramatically with the advent of virtualization. Now, not only services can acquire servers in a datacenter whenever required, but also a server which has available capacity can distribute its resources among a number of services [18] due to virtualization. An extensive account of virtualization has been given in the preceding project work which was conducted in fall 2010 semester [12].

It is the task of management system then to allocate these resources according to the user requests. By the management system, all the resources are viewed as pooled and it can allocate any of them to the users. Hence it has to make decisions for optimal service delivery, for example it has to decide which datacenter to use in order to optimize the service delivery. Also, the already developed techniques of distributed computing are utilized in order to make a service distributed over different clusters and datacenters to perform designated tasks as a single entity [11].

2.1.1.3 Broad Network Access

Another feature required by NIST definition for cloud computing is that the cloud services could be accessible by cloud users wherever they are [16]. This availability should be independent of the physical location, underlying access technology, network provider etc. This has been made feasible because of the rapid advancements in the field of access technology and pervasiveness of computer networks. This also includes that cloud services should be accessible on mobile devices like laptops, smart phones and PDAs etc.

The access network technology has matured both in speed and in pervasiveness [19]. Thanks to the introduction of fiber in access networks, the capacity of access networks is now in Gigabytes per second. Also, the rapid deployment of Wi-Fi hotspots and advancements in Wi-Fi technology has made access networks fast and wide-spread and hence easily available for cloud computing [19]. In addition to the classic way of accessing cloud services, i.e. through computers, some services already have interfaces for mobile phones. For example Dropbox, which is a cloud storage service, has already released an interface which runs on Android running smart phones [20]. This requires that mobile networks should be capable of providing the required speed and with technologies like LTE (4G mobile technology); the downlink speed for mobile network has reached around 168 Mbps [21].

2.1.1.4 Measuring and Billing

As the cloud services are to be provided to the users as utility [1], they need to be properly measured and billed. Just like other utility services, there has to be a mechanism for monitoring the establishment, provision and termination of the service. This metering of the service provision is used by the billing system to bill the user accordingly. The metering and billing systems also have to take into consideration which resources were used, what are the metrics for the resources, how many units of resources were used and so on to come up with a proper bill.

The metering and billing of services provisioned in cloud environment is done by specific elements in the cloud which are managed by the service providers [22]. In our study, we have collectively called all the elements which take part in facilitating cloud service provision under the notion of "management system", as it is explained in Section 2.2.2.

2.1.1.5 Scalability

The concept of infinite resources, as required for cloud computing [17], also calls for rapid scalability. This implies that additional resources are allocated to the service whenever required, automatically [16]. In addition, idle resources are freed for other services to use.

Again, for realization of this cloud computing feature, there must be proper elements and algorithms running in the management system. Hence, suitable management services along with "infinitely" available resources due to virtualization can achieve scalability for services with varying resource requirements.

2.1.2 Cloud Computing Deployment Models

There are three main types of clouds [16] if clouds are classified according to their deployment method.

2.1.2.1 Private Cloud

If a cloud is solely used by a single organization then it is called a private cloud. This includes everything in the cloud starting from the infrastructure to the network internal to the cloud. It is not made available for public to use [23]. It can be owned by the user himself or a third party cloud provider and can be geographically co-located or distributed [16]. Building upon this idea of private cloud, Amazon has introduced Virtual Private Cloud (VPC) service [24] in which the cloud user is connected to Amazon infrastructure dedicated for him through Virtual Private Network (VPN).

2.1.2.2 Public Cloud

A cloud which is available for the general public is called public cloud [23]. The user can access all the services provided by this type of cloud and eventually pay for the services he consumes. The cloud provider is bound to provide a certain level of Quality of Service (QoS) to the user depending upon the SLA they have agreed upon.

2.1.2.3 Community Cloud

This type of cloud is also proprietary but there are more than one organizations with common interests (e.g. security, management, policy, requirement etc.) sharing this cloud [16]. It is again not open for public to use and has all the other properties of a private cloud.

2.1.2.4 Hybrid Cloud

The combination of two or more preceding types of clouds is called a hybrid cloud [16]. The need of hybrid cloud arises, for example, in the case when an organization running a private cloud needs some extra computation resources. Then, that organization can divert some of its computing tasks to a public cloud. There exists an SLA between the two cloud providers to ensure certain level of safety and performance assurance. The two cloud providers remain autonomous and independent entities [16] but manage the flow of data by some standardized technology.

2.1.3 Cloud Computing Service Models

2.1.3.1 Infrastructure-as-a-Service (IaaS)

One of the main services for which clouds are used is the infrastructure provision. The cloud providers own large datacenters which have virtualized servers in large quantities. These datacenters are capable of providing almost any amount of computing required by the cloud user [23]. Hence cloud users can utilize the processors, storage, network or any other type of infrastructure required by them to run their own applications on their own platforms.

The virtualization of hardware owned by the cloud provider has made it possible to dynamically allocate the hardware required by each user when he needs it and readily scale to his requirements. This model thus provides the users control over both the application and the running environment of his application. The most notable example of this service model is Amazon’s EC2 which provides virtual computing environment [4].

The IaaS layer is further divided into sub-models depending upon the type of infrastructure which is provided as a service [25]. Thus, the term DaaS is used to denote Data-storage as a Service while CaaS refers to Communication as a Service. Similarly, the IaaS layer is also divided into Physical Resource Set (PRS) and Virtual Resource Set (VRS) [26], depending upon the control which consumer has over the physical resources.

2.1.3.2 Platform-as-a-Service (PaaS)

The cloud providers also provide platform for development of new applications and services and their deployment in the cloud. So instead of using an application provided by the cloud provider, as in SaaS, the cloud user can develop his own application and run it in the cloud [25]. This service model is called PaaS. In this model, the user has control over the application he is running but still has no control over the platform and underlying hardware provided by the cloud provider [23]. The Windows Azure platform is an example of this service model.

2.1.3.3 Software-as-a-Service (SaaS)

The service provided by the cloud provider can merely be software running in the cloud for fulfilling users' need. It is worth-mentioning that the concept of SaaS existed even before cloud computing emerged [23]. So the initial idea of SaaS was to have an application running over the Internet and it was readily achieved. With emergence of cloud computing, it is now considered as providing cloud services on the application layer.

Thus, in SaaS, the applications are running on the provider's infrastructure and users can access them through web browsers [23]. This has benefited both the users and the providers. On the user side, it has decreased the amount of computational resources required to a large extent. So, even large applications can easily be run in web browsers by utilizing the computational power of the service provider. For the provider, it has made it easier to upgrade, manage and install patches in the applications whenever required without involving the user and asking him to run the updates on his machine.

This model imposes a limitation on the user's access over the source code, but for most of the users who do not tailor their applications, it is not a problem. Besides, the main concern in this model remains the security of user data and availability of the service [25]. So far, these two points of concern are solved using SLAs but the cloud users are still concerned about the degree of reliability on the service providers which keeps the pace of SaaS deployment quite slow. Examples of SaaS, among a lot more, are Google Docs which has built-in word processor, spreadsheet and drawing tools.

The cloud computing models are summed up and presented for better visualization in Figure 1. This figure identifies the two main types of autonomous cloud deployments, public and private, and their aggregation as a hybrid cloud. It also shows the service levels provided by the cloud in relation to their logical distance from the cloud hardware. So there is IaaS closest to the cloud hardware while SaaS closest to the cloud user with PaaS lying in between the two extremes. Some examples of cloud services in each level of service model are also indicated in the figure. The service provider in each service level can be a user of the service provider in the level below it.

2.2 Cloud Management

In cloud service provision, there are a lot of occasions when the cloud is required to carry out some specific management tasks. The nature, scope and level of these tasks differ from each other and hence require an established architecture of the elements carrying out these tasks. In this section a summary of the survey conducted in order to identify the main management tasks and different architectures of management systems is presented.

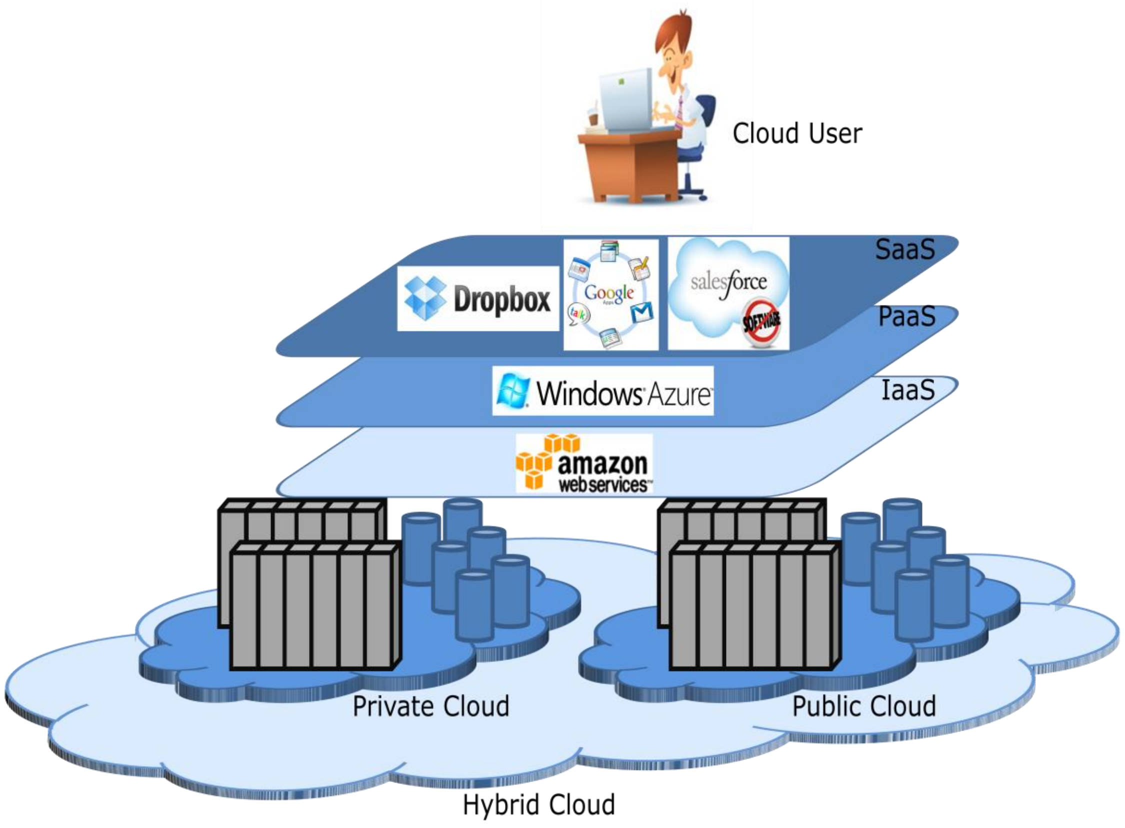

Figure 1: Summary of Cloud Deployment and Service Models [12]

2.2.1 Cloud Management Tasks

In the following, some very important management tasks which are carried out in clouds are enumerated and their brief description is presented.

2.2.1.1 Deployment

The objective of deployment of VMs is to start a new VM at a specific physical server, in a particular cluster of a datacenter. The selection of target physical server is made optimal by considering a number of factors like cost of deployment, load balancing, fault tolerance and power saving.

The decision for placement of VM can be made centrally, an example is pMapper [27] or it can be decentralized, for example using the Ant system [28]. In the former case, the most important factors which are considered are performance, power and cost criteria which the cluster feedbacks to the deployment manager for making the decision. While in the latter case, the most important factors are load balancing and availability assurance for the service.

2.2.1.2 Migration

The process of moving a VM from one server to another is called migration. Nowadays, technologies for carrying out live migrations are also developed and matured in which the VM experiences theoretically zero downtime (actually it is 60 ms) [29]. Migrations are useful for a number of reasons which include; load balancing, power saving and system maintenance [30].

In order to decide when a VM should be migrated and what should be the target for its migration, constant monitoring of clusters and datacenters is required. Thus, there has to be a system which can find the need for migration, determine the target and trigger the migration process. Sandpiper is one of such systems [31]. After it is decided that a VM should be migrated and a target is determined, the migration process starts. There are a number of migration process proposals; two of them are outlined in [29] and [30].

2.2.1.3 Failover

In case of failure of a VM is, there has to be some mechanisms for restarting it again or restore the service by some means. This is called failover. The new VM can be deployed on different levels. It can be in the same cluster, same datacenter or even different datacenter.

There are a number of considerations to make in order to find out where the VM should be restarted. Some of the very common points which should be considered are; the restart latency, amount and scope of data to be transferred [32], effect of the failure of VM on the failover process (for example, considering failures bringing the whole datacenter down) [33] and availability of resources necessary to deploy the VM.

In order to tolerate faults occurring in VMs, which is quite often [34], a number of techniques are employed. Normally, replication of VMs is involved in providing fault tolerance [35]. VMware has introduced fault tolerance on hypervisor level [35]. This means that the management regarding the replicas is done in the virtualization software itself which is installed on the servers and not on higher levels of management system. The layers of management systems according to different criteria are outlined in Section 2.2.2.

Another mechanism required for successful failover is the detection of failures. In most of the cases, a “heartbeat” concept is utilized. This means that all servers send heartbeat signals in order to notify that they are alive (i.e. working) [35] [36]. In case a heartbeat is not received in the specific time, the failover process is triggered.

The failover techniques will be looked upon in detail in this thesis work. This is because this work looks into failures in service provision and their restoration in detail as they form the main constituent in the study of service availability.

2.2.1.4 Accounting

As already mentioned, one of the goals of cloud computing is to provide computing to consumers as a utility and hence a pay-as-per-go model is utilized for the purpose. The details of how this model works is given in [23].

The main idea is to measure the resources utilized by the consumer and bill him accordingly. The billing system gets the metrics of consumption from the metering system and generates the bill. This makes the metering system even more crucial than the billing system [37].

In order to implement the metering system, a number of solutions are developed and commercialized, for instance JXInsight [38]. At large, the different metering solutions for SaaS can be divided into three groups; Non-Intrusive metering (a piece of code is injected in runtime), Intrusive metering (metering code added inline in the business logic) and a hybrid of the two [39].

2.2.2 Cloud Management Systems

As seen in the preceding discussion, there are a lot of management tasks to be accomplished in order for cloud services provision business to run effectively and fulfill its goals. This calls for some management systems to carry out these tasks. So far, there has not been a standard architecture for cloud management system [40] although some proposals exist. In this section, some of these proposals will be discussed. It is worth noticing that different architectural proposals actually aim at managing different levels of cloud computing which are shown in Figure 1. So, there are management service components for each level of cloud service provision and they have different tasks to perform.

On the lowest level of management, that is the virtual machine, there exist different proposals of management system architecture. One of these proposals is derived from grid computing paradigm as cloud computing leverages it. This architecture is based upon the separation of concerns between layers and hence results in two distinct layers for the **management of virtual machines** [40]. The first layer is called the "Management Layer" and is responsible for overall management and performs tasks like decision of where to deploy a VM, control of resources, accounting for usage etc. The second layer is called the "Implementation Layer" and is responsible for tasks like hosting the VMs and maintaining the virtual networks across different datacenters [40] etc. In this approach VMs are described using descriptors and interfaces are made on each layer for different tasks to be performed. Figure 2 shows the way these layers are stacked, some of the responsibilities of each layer and the interfaces among them. One standard fitting this layered model is Open Virtualization Format (OVF) by Distributed Management Task Force (DMTF) [40].

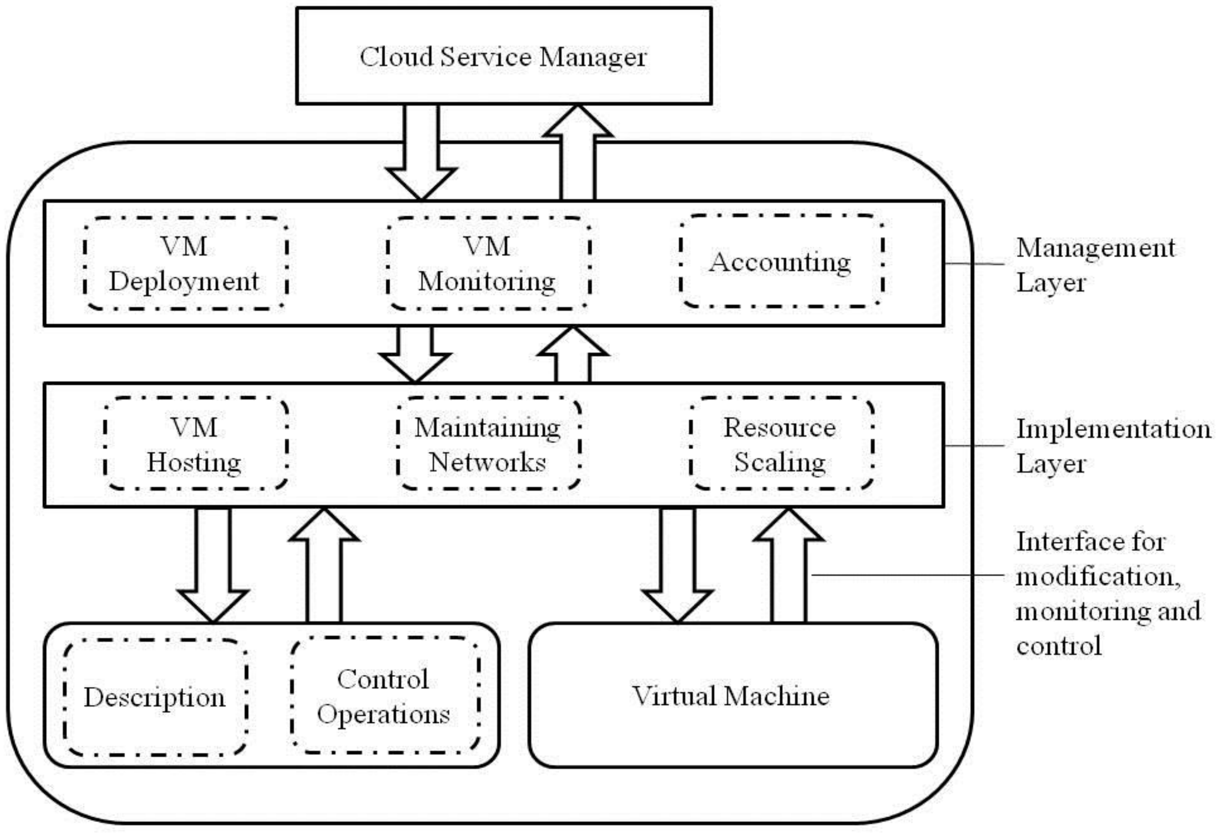

Figure 2: Layered Architecture of VM Management System for Cloud Services

For the IaaS level service provision management, there also exist a few cloud management systems, OpenNebula being one of them. It is worth mentioning and discussing its architecture here because it is open source and also highly deployed [41] and also because its stack is used as a reference for other research projects. Another management system lying on the same level is VMware's vSphere [42]. These management systems fall under the category of **virtual infrastructure managers.** So the main tasks performed on this level include [42] provision of view of the whole shared pool of resources to the user, managing VM's life cycle and scaling the resources for the service automatically whenever required. Figure 3 shows the placing of OpenNebula component in the cloud ecosystem for IaaS service provision.

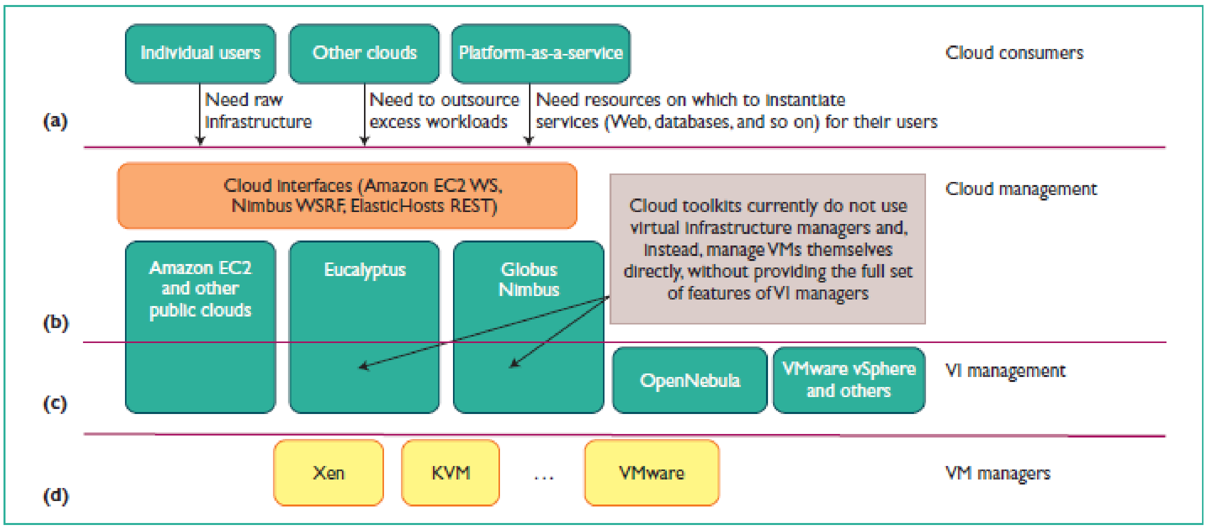
Individual users
Other clouds
Platform-as-a-service
Cloud consumers
Need raw infrastructure
Need to outsource excess workloads
Need resources on which to instantiate services (Web, databases, and so on) for their users
(a)
Cloud interfaces (Amazon EC2 WS, Nimbus WSRF, ElasticHosts REST)
Cloud toolkits currently do not use virtual infrastructure managers and, instead, manage VMs themselves directly, without providing the full set of features of VI managers
Cloud management
Amazon EC2 and other public clouds
Eucalyptus
Globus Nimbus
(b)
OpenNebula
VMware vSphere and others
VI management
(c)
Xen
KVM
...
VMware
VM managers
(d)

Figure 3: Place of Virtual Infrastructure Manager (VIM) in IaaS Cloud [42]

Similarly, there are quite a number of management systems even for higher levels of cloud service provision, i.e. PaaS and SaaS, and at each level they have different tasks to perform. They also exhibit different composition stacks and architectures. The services like Google AppEngine [43] and Windows Azure [44] provide PaaS services and have their own management systems taking care of service deployment and runtime management. Similarly, all SaaS providers have their respective management systems running and managing the services.

A different approach towards management of cloud services is taken by Kaavo. It has been noted in the systems explained above and the others following the same pattern that in order to manage an application deployed in the cloud, the underlying resources are to be correctly managed. The resources to be managed can be physical hardware or virtual infrastructure. But since each application utilizes a number of virtualized resources of a server and each resource has to be tracked and managed, it becomes extremely cumbersome to manage the application by individually managing the underlying infrastructure [45]. Instead, Kaavo proposes an application-centric management system. According to this proposal, the management system can take care of the entire related infrastructure used by the application. It decreases the management complexity and provides an easy accountability for Service Level Agreements (SLAs) also [46]. Thus Kaavo automates the deployment and provides easy runtime support of applications on SaaS and PaaS levels [47].

Figure 4 summarizes the discussion in this section. It shows the different levels of cloud service provision against the corresponding management layer and some examples of currently available management systems in each layer.

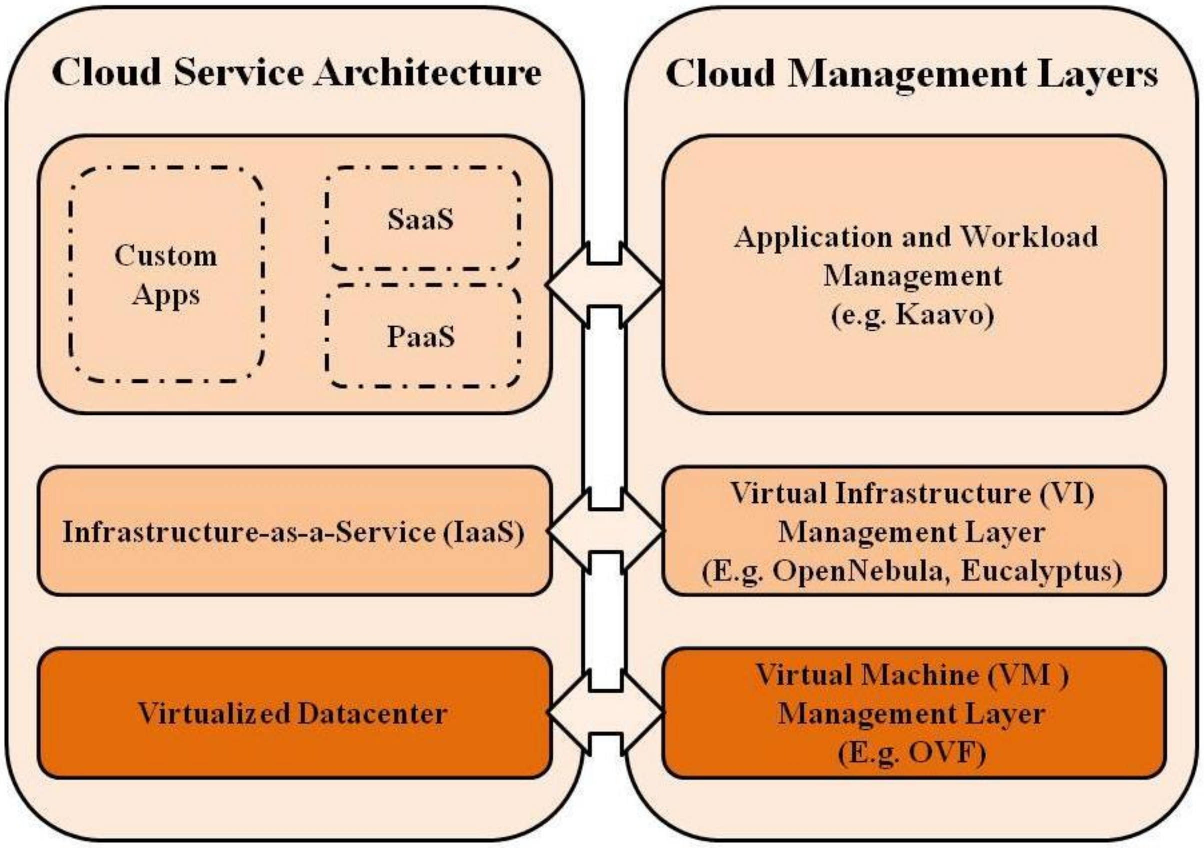
Cloud Service Architecture
Custom Apps
SaaS
PaaS
Infrastructure-as-a-Service (IaaS)
Virtualized Datacenter
Cloud Management Layers
Application and Workload Management (e.g. Kaavo)
Virtual Infrastructure (VI) Management Layer (E.g. OpenNebula, Eucalyptus)
Virtual Machine (VM) Management Layer (E.g. OVF)

Figure 4: Cloud Service Architecture and Corresponding Management Layers

2.3 Classification of Cloud Services

One of the objectives of this thesis work is to investigate different cloud services in the market and find a way to classify them in relation to their dependability. The cloud services can very easily be identified with respect to their deployment and service models but their classification in relation to their dependability requirements is not a straight forward one. It depends upon the criticality of the availability of a service to the user and the extent of provider's investment in resources to achieve a certain level of dependability, among other factors.

This classification will help in generalizing the dependability requirements of different services and in determining the best techniques for ensuring higher dependability for each class. After discussing the cloud computing phenomenon and the deployment and management of cloud services, it is the right time to perform the said investigation and classification.

After looking into cloud services provided by leading cloud service providers, i.e. Google [48], Amazon [49], Salesforce [50] and Microsoft [44] [51] etc, two classification schemes in relation to their dependability are established. The first classification technique takes into account the user's requirement for the dependability of the service and the other technique considers the amount of resources to be provided by the provider in order to maintain a certain level of dependability. In the following, details of these two classification approaches are given.

2.3.1 Customer-Centric Classification (Dependability Requirement Based)

The first approach employed to classify services is by looking at the different dependability requirements the user may has for different services. At this stage, it is instrumental to divide users into two groups, business users and domestic users. We confine our attention on the business users first and domestic users will be dealt with later. For business users, different services have different dependability requirements depending upon the **criticality of the service to user's core business**. The services are decided to be divided into four different groups depending upon their criticality to the user's core business. These classes are shown in Figure 5. The figure also compares these classes in terms of their dependability requirements. Hence it shows that the "Business Core Class" has the highest dependability requirements while the "Value Added Class" has the lowest. Moreover, some examples of services falling in each class are also enumerated in Figure 5.

The criterion of division is as follows. The services are differentiated in two dimensions. The first dimension is the availability requirement of the service. That is, how crucial is the availability of that service to the user. The other dimension is performance. This means that the services are also differentiated considering their performance requirements by the user. The need of two dimensions has arisen from a couple of reasons. First, our definition of service dependability in this thesis work comprises both dependability and performance of the service and hence we are concerned about both. In the second place, the nature of services is different and hence has different parameter of emphasis to be considered as well-performing. For some services, it is very important to have high performance (less jitter, packet loss etc.) for example teleconferencing services. While for some other services it is more important to have high availability, for example emailing service.

One other aspect of this classification is that a single service can fall into more than one classes depending upon the user's usage of the service. This scenario is quite prominent in IaaS and PaaS services where the same service is used for different tasks by different users. So, let's say, a user has procured computing service from an IaaS service provider. Now depending upon the purpose for which this service is used, it can be classified as core business or business support service. Hence, the class of the service to be procured from the provider has to be agreed upon by the two parties and respective SLAs should be made and followed. For this reason, the example services shown in Figure 5 are mainly SaaS and also they are services whose dependability requirements are relatively constant for almost all the users.

A brief description of each of the proposed classes with some example services in each class is given henceforth. This classification is done by looking into the purpose of services, their deployment and their usage by cloud users. The criticality of the service to the cloud user is decided by us and the criteria used are the nature of the service and its target customers.

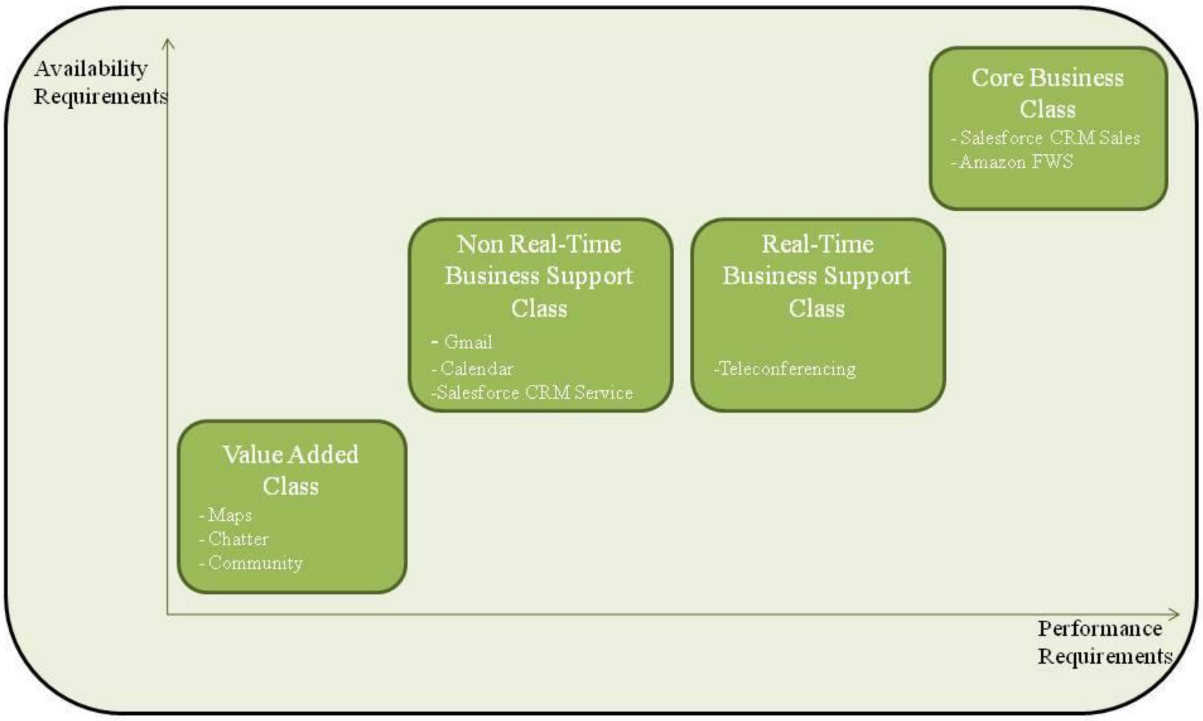

Figure 5: Classification of Cloud Services: Business Users' Availability and Performance Requirements

a. Core Business Class:

This class refers to the services on which the whole or a part of the user's business relies upon. The services in this class are very crucial to the user as it is through these services that the cloud user generates his income. Thus, these services have both very high availability and performance requirements and are least tolerable to any disturbances. One good example of service falling in this class can be a stock market monitoring service hosted in the cloud. This service has stringent availability requirements and also is highly sensitive to delays and losses.

Other examples of services which may fall under this class are; Salesforce Customer Relationship Management (CRM) Sales Cloud [50] and some of Amazon Web Services (AWS) for instance Amazon Fulfillment Web Service (FWS) and Amazon Flexible Payments Service (FPS) [49]. In addition, custom applications tailored by users and hosted in clouds can make Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3) fall in the same category depending upon the criticality of the service to the user.

Salesforce's CRM Sales Cloud is aimed for business personnel in a company. It keeps track of all the data and information about customers and also about marketing campaigns and company sales. So it is an important service for a business because it is through this service and its constituting apps that important business decisions can be made and customers are approached in a satisfying manner.

Also, Amazon's FWS is crucial to its users because it provides a platform for online sellers to sell their goods. Similarly, Amazon's FPS is also crucial to the user because it is used by online merchants to receive the buyer's payment.

b. Real-Time Business Support Class:

The services falling in this class are important for the business users to conduct their business but are not their core business. One important characteristic of these services are their real time requirements. Hence, the teleconferencing services which are used by business users for their internal meetings can be classified under this category.

c. Non Real-Time Business Support Class:

In this category fall the services which have lower performance requirements in the sense that they are not real time but have the same availability requirement as for the previous class. These are the services for which a little bit of unavailability is also tolerable if it is within certain defined limits and don't have as high requirements as the core business services have.

One of the services which define this class is the emailing service. In large businesses, email is the most important and reliable way of communication. Although it doesn't have real-time constraints but its availability has to be high enough for customer satisfaction. Hence Google's Gmail is a candidate for this class. In addition to it, Google Calendar is another service provided by Google which falls under this class [48]. Other cloud providers also have a large deal of services falling under this category, for instance, Salesforce's CRM Service Cloud belongs to this category. In this service, customer's queries are registered and responded to by the company [52].

d. Value Added Class:

In this category fall the services which are used by business users for enhancing their productivity but are not very crucial to them. A large range of services from analysis and maps to chatters and social networks comprises this class. The Google Apps [48] like Analytics, Maps, Chatter and Community etc. fall under this category.

The other classification of the services is done considering the users to be domestic users or small businesses. For these users, the "Core Business Class" is eliminated from Figure 5 and only three classes remain as shown in Figure 6.

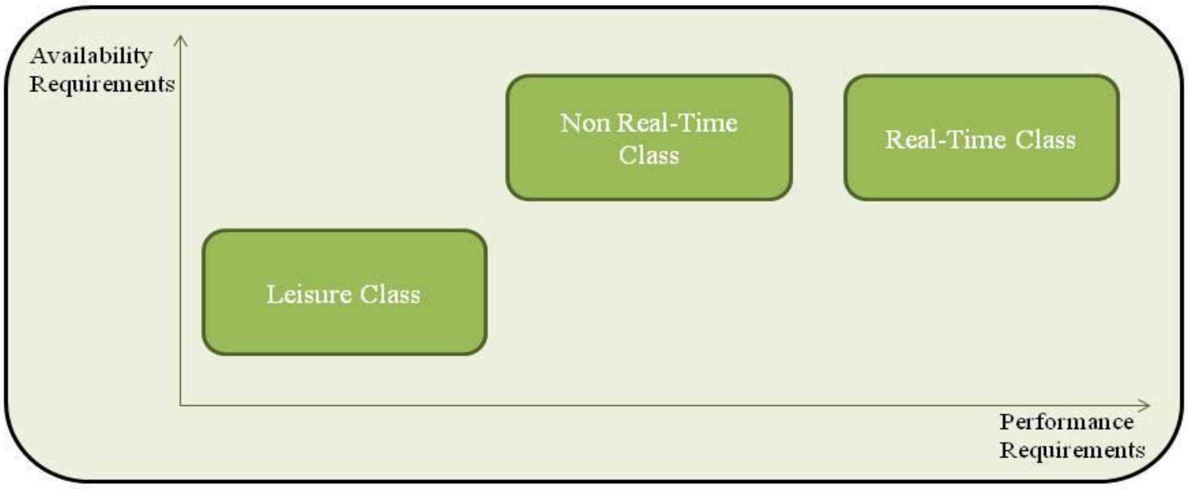

Figure 6: Classification of Cloud Services: Domestic Users' Requirements

The elimination of the high availability class is justifiable due to the fact that the domestic users are considered to be the group of users who don't use cloud services for revenue generation. Thus, the domestic users are considered not to be willing to pay large amount of money for assuring availability. The description of the rest of the service classes for domestic users is the same as the respective classes for business users.

2.3.2 Provider-Centric Classification (Deployment Nature Based)

From the cloud service provider's point of view, the availability of services largely depends upon the resources required for providing redundancy, as it is argued that redundancy is the most common way of achieving high availability [5] [35]. Looking into the cloud services available in the market, for instance Microsoft Azure [51], Salesforce CRM [50], Amazon Web Services [49] and Google Apps [48], it is vivid that different cloud services have different resources of emphasis for their provision. Thus, some services are computing-intensive while others are storage-intensive and yet others are network-intensive. But most of the services provided by cloud service providers are a combination of these types. Hence almost all services, especially at SaaS level, are **aggregated services,** i.e. they aggregate more than one basic service. For this purpose, some aggregator services are also deployed in clouds. An instance of such architecture with basic services and aggregator services is deployed by Amazon [53] and is shown in Figure 7.

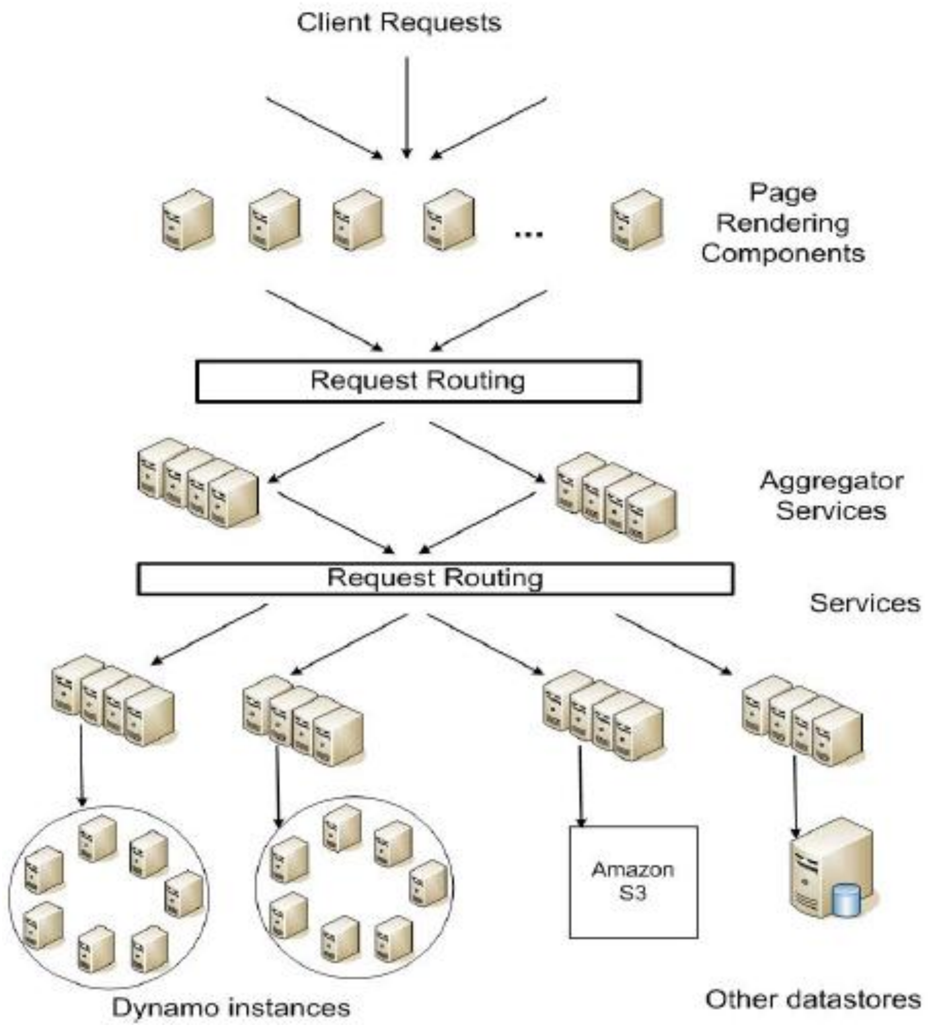

Figure 7: Amazon's Cloud Service Provision Architecture [53]

It is obviously seen from Figure 7 that the results from storage-intensive service "S3" and other services are aggregated by aggregator service to provide final results to the cloud user.

At large, all the services falling under any of the afore-mentioned types can be classified as either stateful or stateless service. The stateless service is the one in which the state of the service is not important to carry out the user request, the aggregator and network-intensive services fall under this category [53] because they do not need to save the state of the service while carrying out an operation. On the contrary, stateful service is the one in which the state of the service is stored and used to carry out an operation by using the service logic [53]. Computing-intensive and storage-intensive services fall under this category. This simple classification can be visualized as in Figure 8.

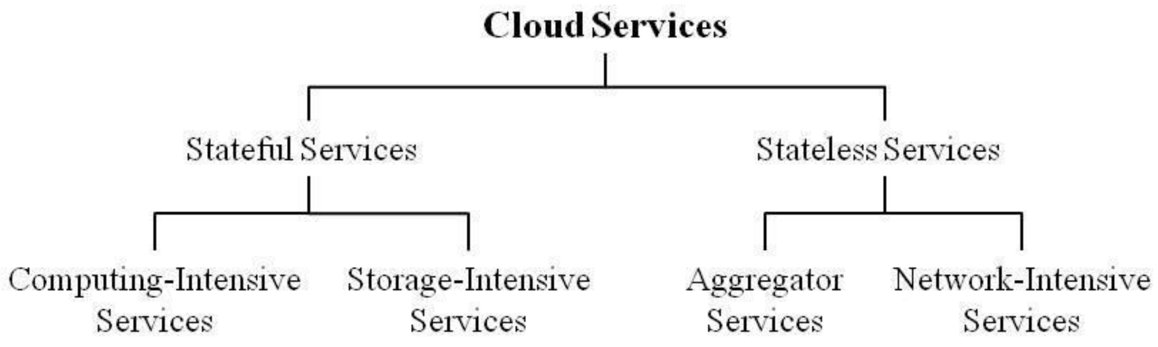
Cloud Services
Stateful Services
Stateless Services
Computing-Intensive Services
Storage-Intensive Services
Aggregator Services
Network-Intensive Services

Figure 8: Classification of Cloud Services: Considering Resources Required for Deployment

The above classification of services considering the crucial resources for their provision gives an insight into two issues. Firstly, it can be used to find the crucial resources for a service and hence redundancy in those resources should be introduced for increasing the availability of the service. Secondly, the service can be stateful or stateless and depending upon this criterion, the type of redundancy scheme can be decided. So, if it is a stateful service, then the state of the replica should be updated from time to time so that it is ready to pick up the tasks in case the primary application fails. But, if the service is stateless then the replica doesn't have to be updated and, in some cases, doesn't even have to be powered on before the failure of the active application occurs.

The study in this thesis work is generalized in terms of resources. This means that we won't consider different types of services with different crucial resources separately. Instead, the study will be general for all types of services. But the statefulness of services remains crucial for the selection of appropriate scheme for enhancing dependability and hence it will be considered throughout the present work. A detailed account of possible techniques for increasing dependability of services with different degree of statefulness is given in the next chapter.

3 Techniques for Dependability Differentiation

In the previous chapter, a brief description of cloud services present in the market is given. These services are also grouped together using different criteria to come up with different classifications. In this chapter, an account of the meaning of dependability differentiation is presented. It is followed by a description of techniques used to achieve dependability in cloud computing. This meets another objective of this thesis work, which is to investigate the possible techniques for providing dependability differentiation in cloud environment.

3.1 Dependability Concepts

In this section, some basic concepts related to dependability are introduced. These concepts are used as basis for conceptualizing dependability differentiation.

3.1.1 Service Availability

Dependability of a system is normally defined as [54] "its ability to deliver service that can be justifiably trusted". Ever since work on dependability of systems was started, a number of attributes has been identified which can quantify the dependability of system in different perspectives. Some very crucial attributes for ICT systems include availability, safety, integrity and reliability.

The phenomenon of performance of systems also lays side-by-side to dependability. If dependability is the ability of a system to provide trustable service then performance is "the ability of a system to provide resources needed to deliver its services" [55]. The common measures of performance hence include carried traffic, congestion, delay and jitter [55].

As it is already highlighted in the previous chapter and is dealt with in detail in the next chapter, there are a number of systems involved in provision of cloud services to their respective users. These systems include cloud infrastructure, management systems, cloud network and access network etc. In order to cater for dependability of cloud service in its totality; a concept of **service availability** is introduced in [11] and is used in the present work also. It is argued in [11] that unavailability and low performing metrics of different systems utilized in providing a cloud service eventually lead to the unavailability of the cloud service if it is stated so in the SLA between the provider and the user. This means that limits of acceptable time delay in network, throughput of cloud infrastructure and all other metrics can be set and not achieving them can be called as service unavailability.

In [11], cloud service availability is said to consist of five parameters namely cloud availability, cloud performance, network availability, network performance and cloud security. As stated and discussed in Chapter 1, only cloud availability and network availability have been accounted for in the present work.

3.1.2 Dependability Differentiation

The notion of dependability differentiation is wide spread in different ICT fields. For instance, it is one of the main concerns in evolution of the future Internet [56]. The main idea is that there is a difference in requirements of technical and economic resources for providing different levels of dependability. It is a known fact that not every user of the same network or cloud needs the same level of dependability in the service he is using. Also, not every application running in a cloud has the same dependability requirements [57]. Hence, providers can guarantee higher dependability to the services and users who are willing to pay for the extra resource investments and go on providing a certain level of dependability to the normal services and customers thus differentiating dependability guarantees for the two. As we are only concerned about cloud availability and network availability at this moment, differentiation in only these two parameters will be considered.

There is extensive work done on the provision of dependable storage systems for cloud datacenters. A statistical study is carried out in [58] on the storage clusters showing the failure trends in them. A step forward towards the provision of differentiated dependability for different types of data is taken in [32]. It is proposed that less accessed data can be stored in fewer replicas compared to the more frequently accessed data which, of course, has higher dependability requirement. Some cost-efficient and automated ways of initiating these data replications are outlined by the same authors in [33] and [59]. There is also some work done on the availability analysis of a single virtualized server [13].

On the network side, quite a large number of well-established techniques for provision of differentiated dependability exist. An extensive survey of these techniques is done in [57]. In their essence, these techniques actually consider the different methods of recovery for different classes of users and services.

3.2 Dependability Techniques in Clouds

One way of increasing the availability of a system is by tolerating faults occurring in it. This means that faults are allowed to occur but their effects are suppressed and do not cause a system failure [55]. Fault tolerance in computing and networking environments is provided by redundancy [35]. This redundancy could be in terms of additional resources, additional mechanisms or just additional time [5]. The most prominent redundancy, however, is additional resources and it is the one which is widely used in today’s cloud environments. Availability enhancing techniques by adding resources are reviewed in the cloud environment in this work.

There are three main types of redundancy [5] for VMs depending upon the state of the replica. In some cases, the replica is up and running and fully or partially synchronized with the active VM while in other cases, the replica is running but only contains a part of the state space. Sometimes, the replica is not even powered on, until the active fails. These three types are called **Hot Standby, Load Sharing** and **Cold Standby** respectively. Thus, there are a number of scenarios for providing redundancy and different scenarios have different implications on the dependability of VMs. Based upon the classification of redundancies performed in [5] and the extent to which states of the replicas are updated, a classification tree of dependability techniques is developed as shown in Figure 9. It should be kept in mind that these techniques are specific to cloud computing domain and specifically target VMs. For the sake of completion, example mechanisms which are already developed employing each technique are also listed under their respective classes.

It is clear from Figure 9 that the replica may either be hot (up and running) or cold (powered off). Next, it is useful to understand the working of hot standby and load sharing replica. In the former case, the standby VM is powered on and running but is not relied upon for providing the output even if it receives all the inputs and performs all the executions, and/or receives state updates from the active VM. On the other hand, load sharing replica divides the tasks with the active VM and a part of input is directed towards the load sharing replica [5]. The service state is updated in both the machines (active and load sharing replica), after designated task is completed by either of them. In case of a failure in either of the two machines, the other running replica takes over the functionality of both replicas until another replica is powered on and updated to share the load with the running replica.

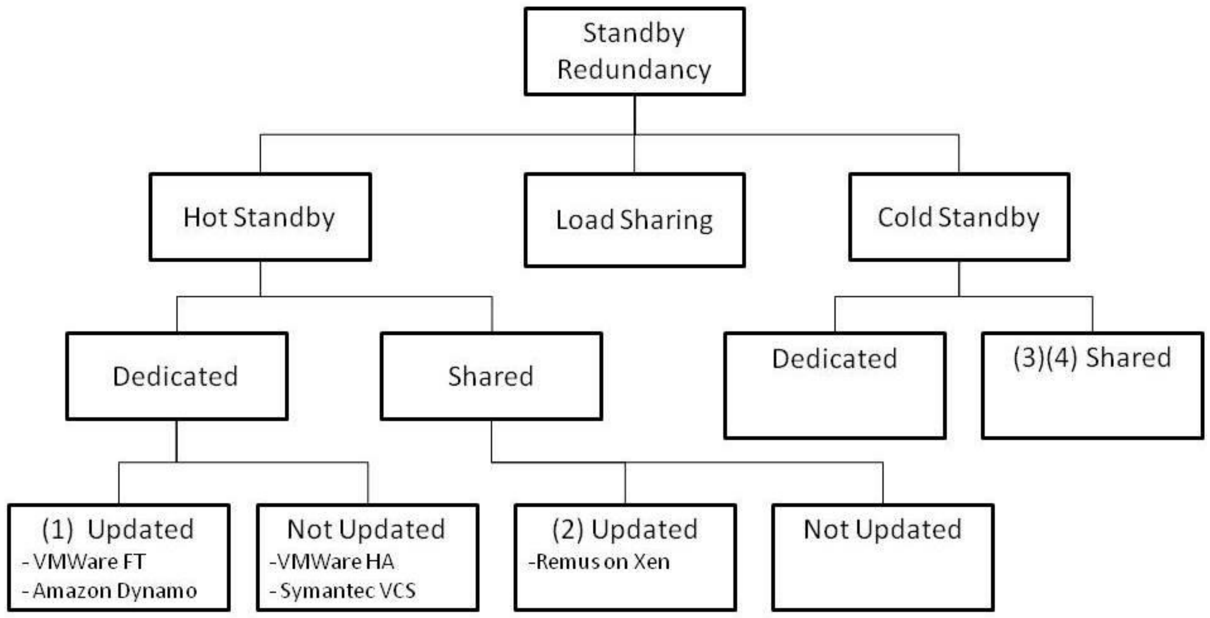

Figure 9: Redundancy Techniques for Providing Fault Tolerance in Clouds

It is worth-mentioning that the notion of dedicated and shared in the classification of redundancy techniques is directly related to the hardware required for deployment of virtual machines. So, for instance, the meaning of dedicated cold standby is that there is hardware resources already dedicated for the replica of a VM although the replica is not running. This will guarantee that when the active VM will fail, the management system will not take time to allocate resources for the replica to run. Instead it will immediately deploy the cold standby using the resources dedicated for it.

On the other hand, sharing of resources in a virtualized system means that the VMs use lesser resources than what they actually require. Thus, more than one VMs can be hosted on the same amount of resources as required by a single active replica. This happens when a standby replica is running and is only getting updates from the active replica. In this way, it doesn't only acquire lesser computing resources but also lesser network resources. Once a standby replica becomes active, it reserves more resources and hence leaves the other replicas with which it shared resources, without enough resources for deployment, in case it is needed.

In order to explain hot and cold standby techniques, some terms are worth defining at this stage. First of them is **crash consistent state**. This refers to the state of a VM from which the VM will be started from in case it fails [36]. Second term of the lot is **Recovery Point Objective (RPO)** which refers to the point in time to which the VM is recovered [60]. RPO is expressed backwards in time from the instant of the failure of VM. Finally, the third term is **Recovery Time Objective (RTO)** and it refers to the duration of time after which a failed service must be restored [60]. RTO and RPO can play important roles in SLAs.

In the following, a brief introduction of examples indicated in Figure 9 of each dependability technique is given very briefly.

a. Load Sharing

In the cloud environment, load sharing technique is considered the one in which both replicas are active but divide among themselves the service requests. After generating an output, they update the service state and in this manner both of them have the knowledge of full service state [5]. In case either of them fails, the other starts responding to all user requests. In this way it is very close to updated dedicated hot standby.

b. Updated Dedicated Hot Standby

The updated dedicated hot standby techniques work in such a way that two totally identical VMs run on different physical servers [35]. These VMs receive service requests and other inputs at the same time and perform execution according to the business logic simultaneously. But, when the output is generated, it is supplied to the user from the active VM only and the output generated from standby VM is suppressed. In this way the states of the two VMs remain intact, synchronized and in case the active VM fails and the standby VM has to take over, it needs no time to catch up. The failure of either VM is detected by heartbeats which each VM sends to the other in order to keep it notified. VMWare Fault Tolerance (FT) is one technique falling in this category [35].

It is obvious from the above description that this type of standby has very high impact on the resource utilization of a cloud provider's infrastructure. It not only has a complete VM running in parallel, and hence claiming its share of CPU, memory etc, but also generates the same amount of traffic to be handled by network inside the cloud datacenter. Similarly, it is as prone to a failure as the active VM as it is running in the same way as the active VM is. This makes this technique very costly, in terms of resources, management and monitoring.

In relation to the services outlined in previous chapter, the stateful services which have very high availability requirement (Stateful Core Business Service) can deploy this technique to meet the user's requirements of service availability.

c. Not Updated Dedicated Hot Standby

In this technique, although the standby VM is powered on and run in parallel with the active VM, it is not actively updated and hence imposes a couple of implications. The first effect is that it uses fewer resources than an updated standby which is positive. But the second effect is that the VM is rebooted on a second server from its crash-consistent state [36]. This introduces some downtime caused by rebooting of VM and updating its state as it falls back in time to RPO.

In case the service using this technique is stateless, there is no downtime incurred because then the standby doesn't need to be updated for state information. But for stateful services this technique introduces some downtime. Thus, it has the same effect on service availability of stateless services as the updated dedicated hot standby has on the stateful services. But it lags behind in providing as high availability to stateful services. Symantec's Veritas Cluster Server [61] and VMWare High Availability (HA) [62] are some products employing this technique.

d. Shared Hot Standby

In techniques falling under this type, although the standby replica of a VM is powered on, it does not use as much resources as the active replica does, even if the replica is updated. The reason is the technology used for keeping the state of the standby VM updated. In Remus proposal [36] which is a mechanism falling under this type, the method used for updating the standby is by check pointing rather than active reception of inputs and actual execution of commands for state update as in VMWare FT. Hence, the standby is updated after certain duration of time by sending synchronization signals from the active VM.

This implies that fewer resources are utilized by standby replica and hence things like CPU, memory etc, are shared by more than one standby replicas. One advantage of Remus proposal is that it updates the standby every 25 milliseconds [36]. This keeps the RPO within a very close vicinity of the failure instant. Also, all the inputs and outputs required to update the state of standby replica from the crash-consistent state to the current state of the active replica are buffered. Thus, in case the active fails, the standby is updated to exactly the same state in which the service was at the time of failure. The frequency of updating the standby determines whether to consider it as updated or not updated, because, if the updates are done rarely unlike in Remus proposal, then it will take long for the standby to take over service provision. The actual downtime in this approach is hence considered to be the time taken by the standby to take over service provision and, in case both replicas fail, the time taken for restarting new VM.

e. Cold Standby

As already stated, the standby replica in this case is not powered on. Instead, it lies in the storage disks of the cloud provider. This replica is switched on and brought to service when the active VM fails. Depending upon the requirements of service users and type of service [5], the crash-consistent state and RPO of this replica is decided. These two parameters will dictate the RTO of the service and hence the duration of downtime.

In Figure 9, although cold standby techniques are divided into dedicated and shared, that is some services may have dedicated resources for their deployment, there isn't any example of this technique found by us in the literature. On the contrary, services with very low availability requirements are not provided hot standbys and hence the management system allocates resources for them whenever it is required [62]. Thus they are classified as having shared resources.

For sake of this study, we have also considered the shared cold standby services to be of two types depending upon their criticality to the users. Hence, we have a higher priority service which is not as critical as to have a hot standby but can preempt some of the services which are even at lower priority than it. This preemption takes place in case there are not enough resources for the higher priority service to restart.

Having seen the different techniques for providing dependability in cloud systems, we hereby limit ourselves to just a few of them for the sake of this study. This is due to the time scope of this study and also to make it precise and compact. There are four techniques chosen for this purpose and are numbered from (1) to (4) in Figure 9. The reason for choosing these techniques is that these techniques cover the whole spectrum of service types we have identified. Thus mappings between these techniques and service classes for both customer-centric and provider-centric can be established.

At this stage we can assume that updated dedicated hot standby can be used in deployment of stateful core business class. Also, updated shared hot standby can be used to provide required dependability to both stateful core business class and stateful business support class. Similarly, shared cold standby can be used while hosting both stateful and stateless business support and value added classes. But these assumptions need to be confirmed by quantifying their availability values. Also the fact that some of them are more resource consuming than the others can affect the choice of most suitable technique.

In the current work we will study stateful services to provide even base for comparison and quantify and compare the availability of the above mentioned techniques in order to find out which technique is best suited for which customer-centric service class. We shall also study the variations in availability figures under different conditions. Then these techniques will be appended with the rest of sub-systems employed in provision of cloud services for studying the effects on higher level of abstraction.

4 Structural Model of System under Study

In this chapter, a system is proposed which will be used to analyze the effect of different dependability techniques in cloud on the offered availability. A high level model of the proposed system is developed in this chapter showing the underling sub-systems employed in provision of cloud services. This model mainly considers the physical infrastructure for provision of cloud services. In order to simplify the model, some assumptions are made but with keeping the model realistic and close to the actual cloud system. These assumptions are also postulated in this chapter.

As the system used in this work is the same as the one used in the preceding work with the same title during specialization project in Autumn, 2010 [12], the cloud sub-systems presented in this chapter are the same as explained in [12]. They are reproduced, rephrased and most of the figures are redrawn for clearing some ambiguities, molding the model in the present study and better presentation. Finally, an account of potential failures in cloud sub-systems which can disrupt service provision is presented the scope of the failures and possibilities of occurrences, repairs and tolerance.

4.1 System Structure

The objective of this project is to carry out a study of dependability differentiation in cloud services and to analyze and compare effects of different dependability techniques. To achieve this goal a service is considered which is running in a cloud. This service is replicated and the replicas are controlled by the management system. Thus, management system decides the number of replicas and if the replica has to be powered on, updated, synchronized and when depending upon the dependability technique employed for the service. Thus, there are different failing scenarios for services employing different dependability techniques. The service failure scenario for each technique will be identified when their availability models will be drawn in Chapter 5.

It is also argued that the service availability also depends upon the location of the replicas, that is, whether they are all located in the same cluster, or different clusters but same datacenter, or different datacenters but managed by the same cloud provider or completely different clouds. These terms are also defined in this chapter before they are used in availability models.

Hence two different levels of differentiating dependability of cloud services are identified. The first level is by using different replication schemes as suggested in the previous chapter and the second level is by spatially separating the replicas.

Figure 13 to Figure 15 show the model of the proposed system which is studied in this thesis work. This model is developed by referring to the physical layout for servers, power distribution and network which is presented in detail in [34], [63] and [19]. The structures presented in the said papers are too large and difficult to be handled in this project, and thus some simplifying assumptions are made in order to limit the model but it is attempted that the model represents the actual system as closely as possible. Therefore, most of these assumptions are mere scale downs. In the following sections, these assumptions along with their justifications, implications and effects appear at appropriate places.

The following section describes the model of the system under study as shown in Figure 13 to Figure 15 in detail and also accounts for inline details of simplifications and assumptions made to come up with the model.

4.2 Model Description, Assumptions and Simplifications

The physical layout of the servers is closely related to the power distribution and network infrastructure. The physical layout as described in [34] and [19] consists of servers mounted in a rack. The racks are arranged in clusters which share the same Power Distribution Unit (PDU) and layer-1 switches. There are a number of such clusters in a datacenter. The foremost assumption made in this regard is thus reducing the actual three-layer structure of a datacenter into two layers. Hence, here-forth, a rack will not be considered as a separate layer, instead a cluster will be considered as the first layer and the datacenter as the second. This assumption is quite natural because the resulting model is not only simple but also actually implementable in small to medium-sized datacenters. Once attributes for this medium-scale datacenter are obtained, the model can easily be extended to datacenters with larger number of layers and its attributes can be calculated. A cloud provider owns a number of such datacenters.

4.2.1 Network Infrastructure

The network in the cloud is classified into four different types [19]. First, the client server network is provided by WAN links. The datacenters are connected to the cloud users through usual commodity networks such as WLAN and Ethernet etc.

Second, the server-to-server network needs additional considerations. The servers located in the same cluster are connected to layer-1 switch and clusters housed in the same datacenter are inter-connected through layer-2 switches. The interconnection of datacenters managed by same cloud provider can have two possibilities. Either they are connected through private network links, e.g. VSAT or Fiber, or they are interconnected through WAN links hosted by independent network providers. In this project, the second scenario will be used because most of the datacenters use this interconnection option along with dual-homing which provides replication at least for the last mile network link entering into the datacenter.

Third, the server-to-storage network which is not considered separately in the current model. Instead the server-to-storage network is considered to be the same as server-to-server network. The reason for this assumption is that the services provided to a user in this study are considered to be either running in a single VM or in case they are running in different VMs then each constituent is considered as a separate service running on a single VM and the service is considered as an aggregated service. This assumption is instrumental in simplifying the model and largely practical as this is the case in delivery of most of the cloud services. But it limits the scope of our study. Thus for keeping the study simple and manageable at this stage, each constituent of an aggregated service will be dealt with as an autonomous service and hence it will be studied thus.

Forth, the management network can be implemented in a couple of ways. It may have a dedicated network or it may be side-banded in the mainstream network. In the model developed in this project, the side-banded management network is being considered. Again this assumption simplifies the model to a great extent and it's practically is unquestionable because most of the SNMP-based management networks are side-banded in the mainstream networks.

In the model, therefore, two layered network architecture is used with all the servers in a cluster connected to layer-1 switch and all layer-1 switches, denoting clusters, connected to layer-2 switch. All the switches are replicated as shown in Figure 10. The links between the switches and from server to layer-1 switch are although replicated but are assumed to be fault-free. This assumption is made because the failures in physical links are very rare compared to the failures in the switch nodes. In case one of the layer-1 switches fails, the other picks up the switching almost transparently. Similarly, each layer-2 switch has two replicas, the standby replica picking up the tasks of the active replica transparently in case the active replica fails.

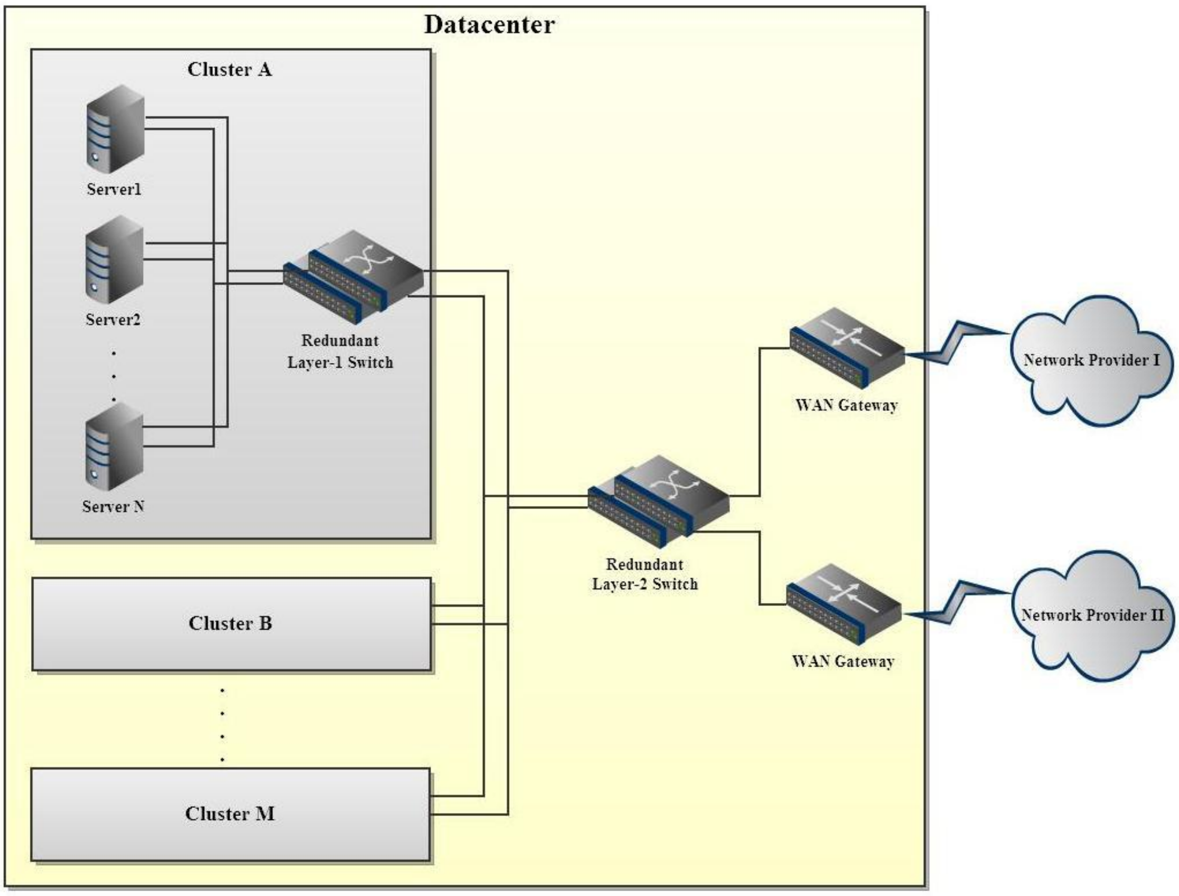

Figure 10: Two-Layered Network Infrastructure

4.2.2 Power Distribution

The physical arrangement of the servers in a datacenter also relates to the power hierarchy of the datacenter. Although there exist a number of power distribution schemes but in this project main motivations are derived from [34], [63] and [64] in which the components used for power distribution and their physical arrangement are very well explained. In the model of the system discussed in this project, not all of the components are explicitly studied. Instead, some assumptions and simplifications are made in order to keep the model simple and easy-to-handle not compromising on their contribution to the availability of the system, though.

The power distribution scheme shown in Figure 11 has considered all the simplifying assumptions and this is the scheme which is used in the system under study. Mainly, a datacenter depends upon the three-phase utility power from local grid station. The power is stepped down at the premises and fed into Uninterrupted Power Supply (UPS). In case of power-cutoff from the local grid station, UPS switches to the standby generators (often referred to as Electrical Diesel Generators, EDG) which are mostly diesel-run. This switching is done by Static Transfer Switch (STS) which is an integral part of a UPS system. There is a small time lag in generator’s taking over the power supply. In this small duration of time (around 15 seconds, [34]) batteries present in the UPS supply the power to the datacenter. Now, the functions associated with all the components mentioned so far are accumulated in the UPS block of Figure 11. Hence the stepping down function and the switching function are aggregated in the UPS block along with its own function of feeding power to the shared power bus which distributes power in the datacenter.

As stated earlier, there exist a number of ways in which the servers can be connected to power infrastructure but in this report, the one described in [34] is used. This structure is classified as Tier II in [64]. According to this structure, the UPS is duplicated and both of the UPSs feed a shared power bus which distributes the power to individual Power Distribution Units (PDU). In the system under study, one cluster is powered by one PDU, although there are individual circuit breakers for each server on the PDU. These breakers are not shown in Figure 11 explicitly; instead they are assumed to be implicit in the servers themselves and hence form a part of power block in Figure 12. This is also reasonable because their failures will be accounted for as server failure due to power-cutoff and this will make no effect on the final result. Once again the PDUs are duplicated and share a bus for feeding servers which is assumed to be never-failing due to same reason as the network links. Layer-1 switches are powered by respective cluster PDUs while each of layer-2 switch and WAN gateway has a separate set of duplicated PDUs for them.

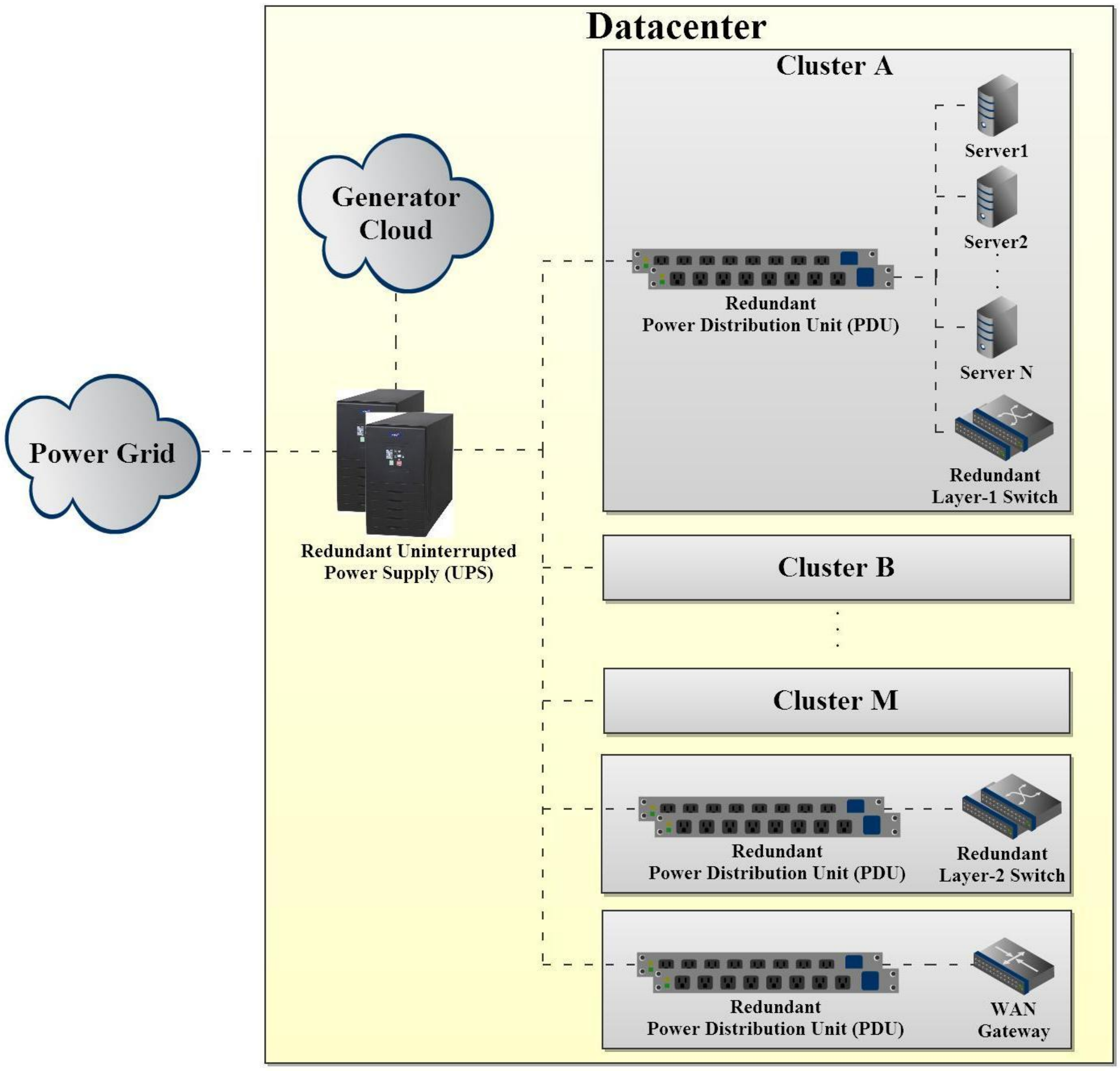
Datacenter
Cluster A
Server1
Server2
Server N
Redundant Layer-1 Switch
Redundant Power Distribution Unit (PDU)
Generator Cloud
Power Grid
Redundant Uninterrupted Power Supply (UPS)
Cluster B
Cluster M
Redundant Power Distribution Unit (PDU)
Redundant Layer-2 Switch
Redundant Power Distribution Unit (PDU)
WAN Gateway

Figure 11: Power Distribution in a Datacenter

The cooling infrastructure for servers is also very vital in large population of servers. The cooling structures in datacenters are outlined in [34] [19]. In this project, a centralized cooling infrastructure is considered. It is powered directly by the UPS and has outfits in all the clusters maintaining a certain level of temperature required by the servers.

4.2.3 Server

Servers are another important block in the model of the system under study. In the virtualized domain, it is logical to deal with physical server separately from virtual server. A physical server accommodates more than one VMs. In this regard, a number of virtual servers utilize the same physical resources. The main components of a virtualized server are [13]; CPU, memory, NIC, power supply, hard disk (HDD) and cooling system as shown in Figure 12.

Host OS can also be considered as a part of physical server in addition to the virtualization layer if the virtualization is considered to be “hosted” (i.e. virtualization layer is hosted by the OS running on physical server), refer to [12]. But for the sake of this project, the virtualization is considered to be “bare-metal” (i.e. virtualization layer runs directly on physical hardware of the server) and hence only virtualization layer is considered to be a part of physical server. Each VM is considered to have its own OS. This choice of bare-metal virtualization of servers is also justifiable considering that the leading manufacturer of virtualization software, VMWare, uses the same technology [18]. The virtualization layer functionality is provided by Virtual Machine Monitor (VMM) [18] and it is also considered as a part of physical server in the model under study [13].

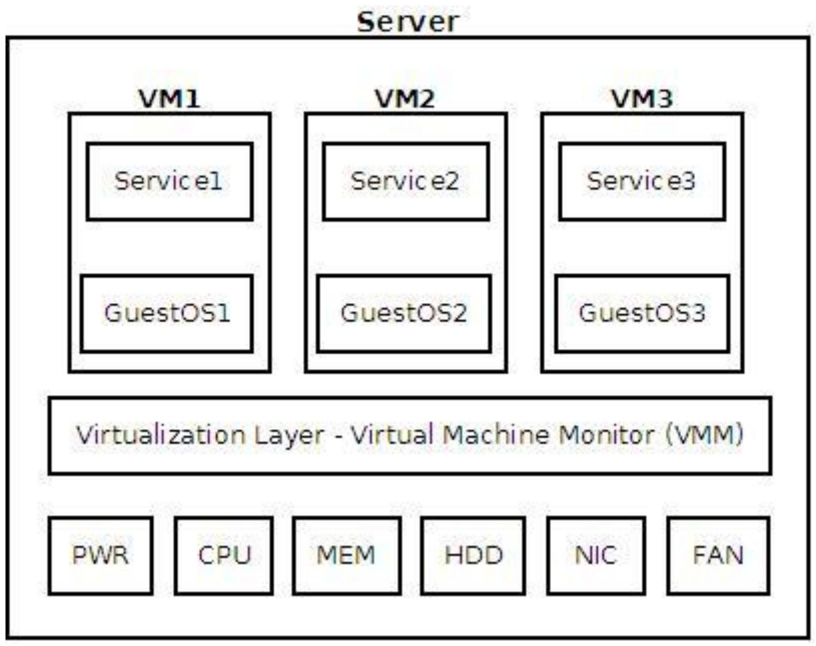

Figure 12: Layered Model of a "Bare-Metal" Virtualized Server [12]

In the presence of virtualization in the servers, it is possible to have two replicas of the same service running in the same server independently. Although possible, this idea is never utilized because it will decrease the independence of failures from each other to a large extent due to a large set of shared failing points such as physical hardware and virtualization layer. Thus all the commercially available solutions which run hot standby replicas do it on different servers in the same cluster or even in different clusters [61] [62].

4.2.4 Final Model

In light of preceding description and assumptions, a detailed model of the proposed system which is to be studied is developed. In order to capture all the details on different levels, it is presented in steps in Figure 13 to Figure 15. The first level is a cluster of servers as shown in Figure 13, the second level is a datacenter with a number of clusters in it, shown in Figure 14, and the third level is a cloud with a number of datacenters which are geographically distributed as illustrated in Figure 15.

Figure 13 is self-explanatory, it clearly shows the replicated network links and the shared power bus as well as duplicated PDUs and layer-1 switches. It defines a cluster to be an array of servers powered by a single PDU and connected to a single layer-1 switch. Also there are management systems which run on cluster level and provide management of VMs hosted in a single cluster. One such management system is Veritas Cluster Server (VCS) which is developed by Symantec [61].

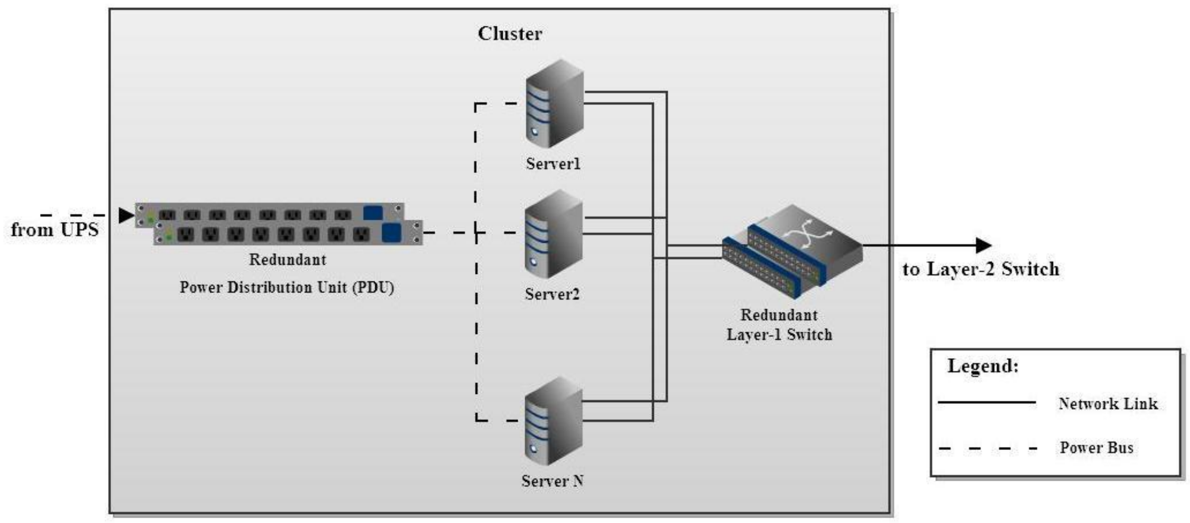

Figure 13: A Cluster of Servers

In Figure 14, the components of datacenter as used in the system under study are identified. It shows the presence of centralized cooling system for a datacenter. As already mentioned, the cooling infrastructure is independent of power distribution system. It is powered by UPS and has outlets in different cluster rooms as per the requirements. In this study, although it is included and accounted for, it is not dug into deeply.

The use of word “cluster” for both switch and router in Figure 14 denotes the presence of PDUs for each of them; this is illustrated in Figure 11.

Finally, Figure 15 shows two cloud providers. It is worth-noticing in Figure 15 that not only two different cloud providers are connected via WAN links and independent network providers but also different datacenters of the same cloud provider are inter-connected by the same method. The figure also shows that the connection existing between the cloud user and cloud provider is also a WAN connection which passes through the same core network as the interconnection of datacenters. The core network here refers to the backbone network. For the case of Norway, for instance, it is the Norway Internet eXchange (NIX) supplemented by the local Internet Exchange of a city.

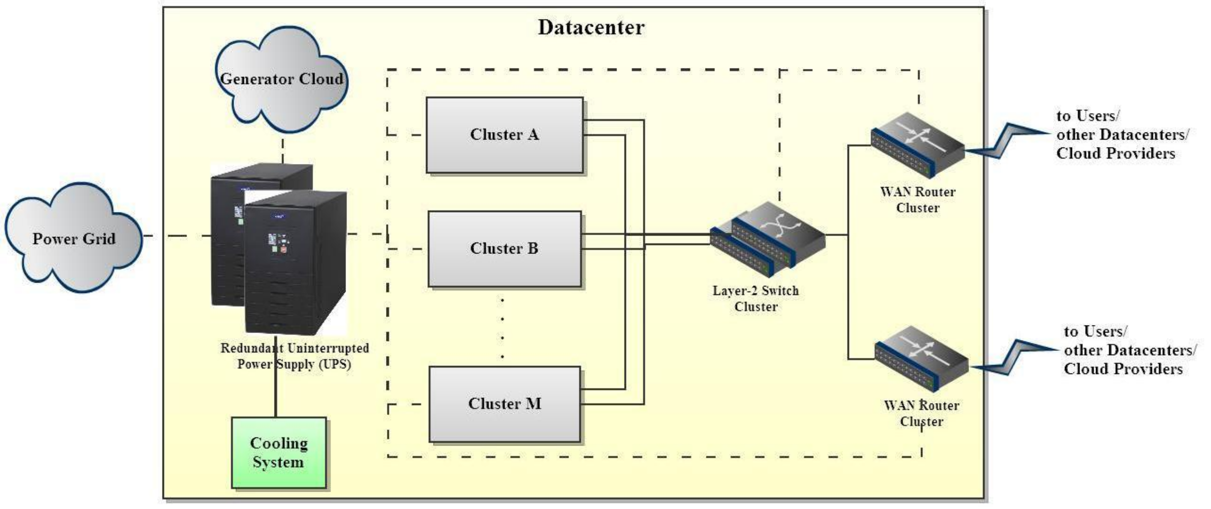

Figure 14: A Cloud Datacenter

Figure 15 also shows that the network provider is replicated. This actually means that two independent network providers are employed to provide WAN services in such a way that one of them back-ups the other. Although load balancing can also be employed in this scenario but in this project it is assumed that it only provides backup for the primary path. This assumption is also worth-doing because this scenario is cheaper than the former due to the reason that the data flow through the backup path is quite rare (only in cases when the primary link is down).

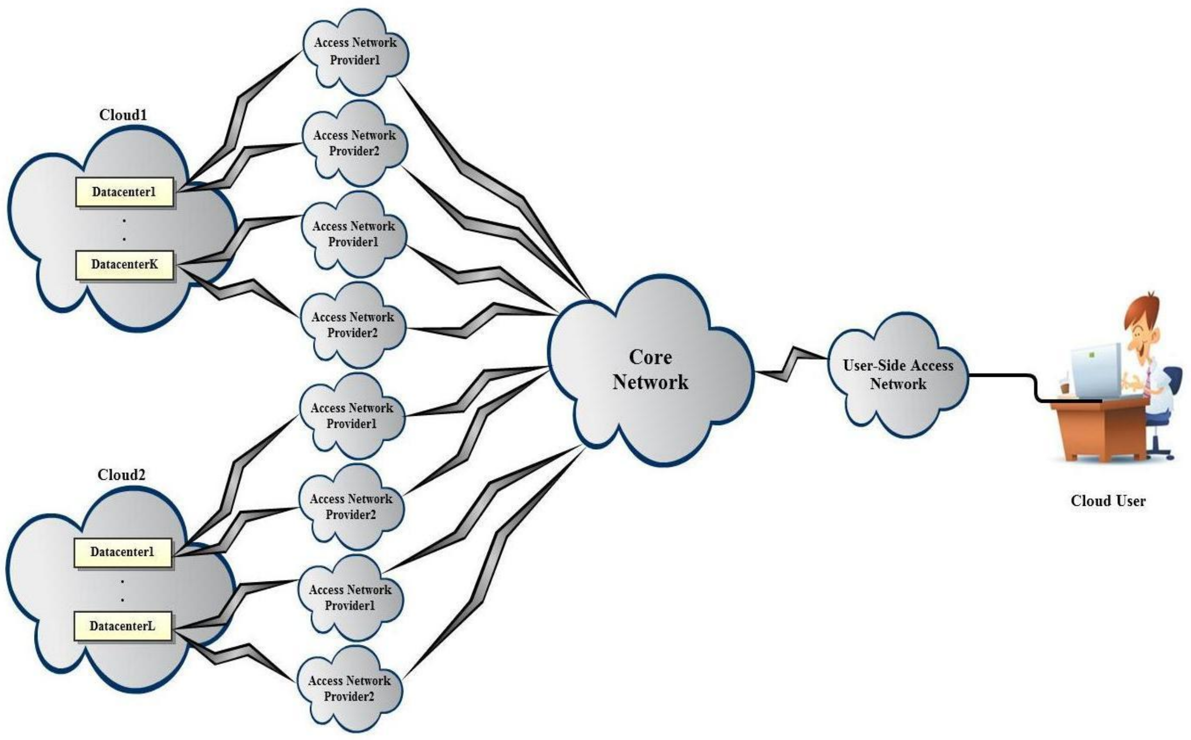

Figure 15: The Cloud Providers

Thus, a model for the system under study is developed, described, commented on and presented in figures in this section.

In the next section a brief account of failures which are studied in this work is given. Although there are a number of different types of failures in ICT systems but in this study only the most common and favorable to occur failures are studied. Later on, the service availability will be calculated by considering each of these failures. Hence it is important to outline them, understand their cause and comment on their scope, their remedies and mechanisms for their tolerance.

4.3 Failure Classification

In a cloud system, there are a number of failures in individual components which have a potential to bring the service down, or at least to affect the service provision to some extent. In the following these failures are classified according to the causes of these failures. The causes are grouped in line with descriptions in [55] [5].

4.3.1 Hardware Failures

Hardware failures are normally caused by physical faults. Physical faults are also known as solid faults. These are the classical faults occurring due to wearing out of a component, degradation of a device with time or due to over-stressed power, pollution or mechanical stress. These faults are permanent in the sense that they persist until repaired manually by replacement of component or some other manual activity [55].

In the proposed system, there are two sources of hardware failures which are; the cloud and the network. There are a number of different types of physical failures with different scopes and effects in each of the two sources. To start with, the most commonly occurring hardware failure in cloud is hard disk failure [65]. Around 78% of hardware failures in Microsoft datacenter are hard disk failures [65]. The wear of hard disks in cloud servers is also higher compared to the stand-alone servers due to the higher rate of utilization of the cloud servers [58]. The higher utilization of cloud servers is direct consequence of the virtualization. The same phenomenon also holds for NIC cards, memory, and all other nodes as highlighted in different researches carried out in this regard [65] [58].

The scope of a hard disk failure is limited to a physical server but consequently it affects more than one virtual machine running on the same server. Hence all the VMs running on a single server will be affected by hard disk failure. The other prominent failures in clouds like RAID controller failure, memory failure and other component failures as outlined in [65] also have the same scope. The remedy for such failures is replacement of the component and it takes some time which introduces a downtime, which can be tolerated using FT techniques. These techniques cause the downtime to decrease considerably and provide room for repair and maintenance without experiencing any downtime.

In the network side, the hardware failures are due to weary network nodes; i.e. switches, routers, gateways, links and NICs of the servers. Again it is argued in [58] that higher link utilization increases the failure rate of links and nodes in the network in clouds. The utilization of link increases because of the deployment of more than one virtual server on a physical server; this causes greater data transfer among servers hence increasing the link utilization. The scope of such failures is not limited to a single server, it can bring down a whole cluster and even a whole datacenter depending upon which component has failed. For instance, if just a link from a switch to a server is failed, it will cause that server to be isolated from the rest of the network. But if the WAN gateway fails, it will cause the whole datacenter to be disconnected from the Internet and hence affecting the availability of the whole datacenter. The remedy of solid failures in network is, like other hardware failures, to replace the faulty component.

Due to the high dependence on network for service availability, the links, switches and even WAN links are replicated. The replicated switches provide uninterrupted connection within the datacenter and the replicated WAN links, normally dual-homed, account for datacenter availability even in the case when one WAN link is not functional.

4.3.2 Software Failures

Software failures are mainly caused due to design faults. These are human made faults which affect the logic of the system [55]. Design faults are also called logical faults. This type of fault covers a vast arena starting from basic design flaws to implementation failures. The examples of design faults may include lack of timing and synchronization as well as inconsistent system specification.

Software faults are a sub-class of design faults and are also known as “bugs”. A distinction between reproducible and irreproducible bugs is found in literature, e.g. in [55] and [5], and the respective nomenclature is that the former is called “Bohrbug” while the latter is known as “Heisenbug”.

It is seen in [34] that about 35% of the total downtime in Google datacenters is caused by software failures while only less than 10% is due to hardware failures. The reason for such a low downtime due to hardware failures is the extensive deployment of fault-tolerance techniques. Downtimes in software, on the other hand, remain short but frequently occurred. The software failures can affect the cloud system on a number of levels. They might just affect a single VM if there is a fault in VM software. The effect may spread on a number of VMs running on the same physical server if the fault happens to occur in virtualization layer software (VMM) running on the server. The extent of fault effect will be as vast as the whole cloud if the cloud management software happens to have software failures in it. In addition to software failures occurring in the cloud, even networking devices are vulnerable to software failures. So a software failure in a networking device affects the cloud system on the same level as the hardware failure in that device but with reasonably shorter period of downtime.

The removal of “Bohrbug” is highly recommended as it is easier to detect, since it reproduces under the same conditions. The way to repair this failure is by removing the bug from the code itself [66]. The most extensively used fix for “Heisenbug” is by restarting or rebooting the application. In some cases replicating the system on other servers is also helpful, [66]. In the case of software-aging, rejuvenation is proposed in [66] to keep the system up and running for the most of the time. Rejuvenation is the phenomenon of restarting the system after a defined interval of time in order to prevent occurrence of a failure. This interval is found by experience with the particular system and can be different for different systems.

4.3.3 Operational Failures

Referring again to the data provided by Google [34] it is seen that about 30% of failures in datacenters occur due to operational and configuration faults. These are accidental faults made by human operating or configuring the system, either for update of the system or during a repair. The extent to which this type of failure affects the cloud system depends upon the level on which the fault has occurred. It might affect only a single VM if the fault occurs in virtual system software. It may affect a physical server, thus affecting all the VMs running on it if the fault occurs on virtualization layer. There is a possibility of affecting a whole cluster or even a whole datacenter in case network node software is mis-configured. The worst case, however, remains the mis-configuration of the cloud management software which can bring down all the cloud at once.

It is argued in [67] that most of the operational faults occur during routine maintenance, i.e. upgrading hardware or updating software patches etc.

4.3.4 Environmental Failures

Environmental disasters also play a part in the dependability of a system. Factors like floods, power outages, fires etc. are although outside the control of the cloud service provider but can always interrupt service provision, and mostly to a large extent. This is because factors like floods and power outages affect a whole datacenter and hence their consequences can be very large-scale service disruption.

The functionality of servers also depends upon the thermal conditions of the place where the servers are installed [19]. Hence failure in air-conditioning system of the premises where servers are placed also causes failure in provision of service by these servers. So the servers will either shutdown completely or will under-provide the service and hence can be regarded as unavailable. The severity of these failures also varies. It might be a fan of a single server not working hence causing only one server to be affected, or it might be failure of the cooling system of a cluster room where servers are installed which will eventually cause all the servers in that cluster to become unavailable.

In addition, power disruption in datacenters also has potential to effect the service provision of a datacenter. In light of fore-mentioned power distribution infrastructure, it is deduced that there are a number of failure points which affect the service provision on different scales. First of all, if power supply is cut from the electric company then there is a potential of whole datacenter going down but normally it doesn’t happen due to UPS taking care of this scenario. But then, UPS can go down itself or EDGs may fail to start and hence can cause power outage on datacenter level. Another point of failure is PDU whose failure will only affect the cluster to which it is supplying power.

In the discussion above, an account of approximately all the possible failures in clouds and networks leading to failures in cloud services, is given. In Table 1, these failures are listed and their extent of service disruption is indicated. Also the methods of their remedy and tolerance are indicated. One interesting failure cause is the management system. The management system exists for a cluster, a datacenter as well as for a cloud provider. In case of a service running on different clouds, a management system is also required for the service as a whole. In the former three cases, the management system is usually maintained by the cloud provider but in the latter case, the management can be done by a user side software or a third party which provides the hybrid to the user. In this case the hybrid implementation will be transparent to the service user.

Table 1: Summary of Potential Failures in Cloud Services [12]

Failure		Extent of Service Disruption					Remedy	Possible Tolerance Tehcnique
		VM	Server	Cluster	DC	Cloud		
Hardware Failures	Server Components	Yes	Yes	No	No	No	Replace	Replicated VMs in other servers
	L1 switches	Yes	Yes	Yes	No	No	Replace	Replicated switches
	L2 switches	Yes	Yes	Yes	Yes	No	Replace	Replicated switches
	Routers & WAN Link	Yes	Yes	Yes	Yes	No	Replace	Dual-homing
Software Failures	L1 switches	Yes	Yes	Yes	No	No	Remove fault/Restart/Reboot	Replicated switches/Rejuvenate
	L2 switches	Yes	Yes	Yes	Yes	No	Remove fault/Restart/Reboot	Replicated switches/Rejuvenate
	Routers	Yes	Yes	Yes	Yes	No	Remove fault/Restart/Reboot	Replicated switches/Rejuvenate
	VM (Application + OS)	Yes	No	No	No	No	Remove fault/Restart/Reboot	Replicated VMs in other servers
	Virtualization Layer (VMM)	Yes	Yes	No	No	No	Remove fault/Restart/Reboot	Replicated VMs in other servers
Power Outages	UPS	Yes	Yes	Yes	Yes	No	Replace	Replicated VMs in other datacenters
	PDU	Yes	Yes	Yes	No	No	Replace	Replicated VMs in other clusters
	Sources (Utility Power Supply, EDGs, Batteries)	Yes	Yes	Yes	Yes	No	Repair	Replicated VMs in other datacenters
Manage-ment System	Cluster (VM Management)	Yes	Yes	Yes	No	No	Remove fault/Restart/Reboot	Replicated VMs in other clusters
	Datacenter (VI management)	Yes	Yes	Yes	Yes	No	Remove fault/Restart/Reboot	Replicated VMs in other datacenters
	Cloud	Yes	Yes	Yes	Yes	Yes	Remove fault/Restart/Reboot	Replicated VMs in other clouds
	Service	Yes	Yes	Yes	Yes	Yes	Remove fault/Restart/Reboot	
Operational Faults		Yes	Yes	Yes	Yes	Yes	Remove fault	
Natural Disasters		Yes	Yes	Yes	Yes	No	Repair Infrastructure	Replicated VMs in other datacenters
Cooling System		Yes	Yes	Yes	Yes	No	Repair	Replicated VMs in other datacenters

5 Analytical Models: Design, Analysis and Results

In the previous chapters, we have sketched a model for cloud service provision system and explained its constituent sub-systems. We have also stated the assumptions and simplifications made in the model in order to keep it close to reality but at the same time possible to handle in the present work. Failures in cloud systems are also detailed and their extent of effect on the service provision is explained.

In addition to this, cloud services are also classified employing two different classification schemes. Also, different techniques for enhancing service availability of cloud services are found from the literature. It is assumed that different services will meet their dependability requirements by employing different techniques depending upon their nature of deployment. Then again, the techniques for guaranteeing dependability to different services are divided into two levels; depending upon the state of standby replica and depending upon the physical location of the replica.

In order to study the effects of these techniques on dependability of cloud services which are hosted in a cloud system outlined in Chapter 4, some analytical models are drawn. These models are presented in this chapter. Moreover, some assumptions are made in order to use standard dependability models; reliability block diagrams and Markov models to be precise. These assumptions are listed in Section 5.1.2 for each model.

In addition, it is noted that most of the models can be simplified and some of their constituent components can be dropped as their contribution to the total service availability is negligible compared to the other components. These simplification possibilities are exploited, implemented and validated in Section 5.1.3. The examination and simplification of models is also one of the main objectives of this thesis.

5.1 Modeling Framework

In this section, the approach used to model different dependability techniques on different levels (refer to Section 4.1) is given. It is then supplemented by assumptions made for model selection and also methodology for simplification of different types of models.

5.1.1 Model Selection

As already stated in Section 4.1 that there are two different levels of dependability differentiation, two different levels of models are also developed. Therefore, reliability block diagrams are used for modeling the differentiation considering the spatial separation of standby replicas (henceforth called **Cloud Level Model**) while Markov models are used for modeling the differentiation depending upon the state of the standby replica (henceforth referred to as **VM Level Model**).

The reason for the former selection is that reliability block diagrams take into consideration the structure of the system. In the current study, it will help because in order to model spatial separation, the structure of the system has to be considered and the placement of the replica has to be shown in the model. Nevertheless, there are some assumptions made in order to use reliability block diagrams for this purpose and they impose some limitations, which are looked at in Section 5.1.2.

The reason for the latter selection is that in order to model states of active and standby replicas, we need a model which can model states and transitions between the states following some events. Reliability block diagrams can't model states but Markov models can, and hence they are used for this purpose. Again, there are assumptions made in order to use Markov models, listed in Section 5.1.2, for the said purpose, but it is clearly seen that the limitations imposed by Markov models are less limiting than reliability block diagrams. The freedom of modeling the state of the system and change in its state in response to some events gives dynamicity to the model compared to the static model resulting from reliability block diagrams. So, although each of them has its advantages and disadvantages, each of them is selected for a selected purpose due to a specific reason as already mentioned.

5.1.2 Model Assumptions

In the following, the assumptions made in order to use a reliability block diagrams are postulated [5]:

1. The components of the system (blocks in the diagram) fail independent of each other. In reality this assumption is not very practical because a number of failures can occur depending upon other failures.
2. The components that fail are restored independent of each other. Again, this assumption is quite impractical because it requires a dedicated repairman and test equipment for each system component.
3. The system is assumed to work as it is intended. This means that fault tolerance will provide service as long as there are resources available in the resource pool. This assumption is also very idealistic because in reality the efficiency of most of the systems is less than 100%.
4. In addition to the above assumptions, one other shortcoming of reliability block diagram is that it can't be used to model state of the system. So if, for instance, there is a delay in reboot and the system is in transition state, it can't be shown in a reliability block diagram.

Similarly, there are also assumptions made for using Markov model for modeling dependability techniques. These assumptions are listed as follows [55]:

1. In order to develop a Markov model, it is assumed that the inter-failure time of the system is negative exponentially distributed (n.e.d.) with the parameter given by the failure intensity, $\boldsymbol{\lambda}$ in this study. Although most of the IT systems and components in the real-world obey this assumption [55], it remains a very idealistic assumption.
2. The time between repairs is also assumed to follow n.e.d. repair intensity, denoted by $\boldsymbol{\mu}$ in this work. The impact of this assumption is the same as mentioned above for failure intensity.
3. The succeeding sequence of events (whether failures or repairs) does not depend upon the time spent in the current state. This is a direct consequence of the memoryless property of n.e.d.
4. The next event can easily be determined when the active processes are known. This is also due the property of Poisson processes that two independent processes can be merged by adding their intensities. Similarly, the probability of an event occurring first is found by dividing its intensity by the intensity of the merged process. This assumption also limits to some extent the perfect randomness which the event occurrence has in reality.
5. Markov model doesn't take the structure of the system into account. Instead it only depends upon the states of the systems and transitions between them. This leads to rather complex models, even for less complex systems.

5.1.3 Model Simplification (Parsimony)

As already stated, the models can be simplified and some of their components/states can be dropped depending upon their contribution to the final value of service availability. This simplification will cause the service availability to be approximated by fewer parameters, also known as **parsimony**, and the models will be **parsimonious**. This process of simplification is made possible by solving the models, finding the parameters which have least effect on the total availability and dropping them. This approximation is further validated by using the actual values of parameters as found from the literature and listed in Table 3 to see if the parsimonious models reflect the original models very closely.

In order to come up with parsimonious models of reliability block diagrams, the models are solved for determining the service availability expression. Then, the components which contribute the least to the availability are tried to be dropped if it is justified that other components' contribution is very high compared to them and by dropping them will not have much effect on the final expression for service availability. This parsimony is then validated by using numerical values from Table 3.

In case of Markov models, the values of intensities and probabilities that mark transition probabilities are analyzed. This gives an indication of the possibility for the system to be in a certain state. A state which is found to be very unlikely to be taken by the system is discarded. The parsimonious model is then validated against the actual model by using numerical values for the transition parameters and comparing the results obtained from the two corresponding models.

5.2 VM Level Models

In this section Markov models for different dependability techniques depending upon the state of the standby replica are drawn. These models are then solved by using packages developed in Mathematica 6.0 by B. E. Helvik [68] for solving Markov state diagrams. In the next step, states which are very unlikely to happen are identified and discarded from the models and the result is compared to the respective original models. In this way, the models are made parsimonious, wherever possible, so that it is easier to handle them later on. In relation to the techniques identified and selected in Section 3.2, four distinct models are developed.

The services employing different techniques are also prioritized in a way. This is because if a user is paying for, let's say, updated hot standby replica (which is the most expensive as it utilizes the most resources), then he needs to have highest possible availability. The prioritization takes effect when there is a competition for reserving resources. Hence, in case of scarcity of resources, the service with the highest priority will be allocated the available resources. The priority order is given below:

1. Updated Dedicated Hot Standby
2. Updated Shared Hot Standby
3. Cold Shared Standby (High Priority)
4. Cold Shared Standby (Low Priority)

5.2.1 Updated Dedicated Hot Standby

In the following the details of model development for this technique are given. It is incorporated with inline assumptions and descriptions of different states and transition parameters.

As described in Section 3.2, there are two identical VMs running parallel to each other in this technique, hence the factor "2" is multiplied with failure intensity **λ** in Figure 16 as both of them are equally prone to failures. There is always a third cold replica of the active VM which lies in the storage of cloud service provider. As argued in Section 3.2 under the item "Cold Standby", it is the RTO and the RPO of the service that determines how updated this cold standby is. In this case, we assume that the cold replica lying in storage is updated to a very close point of time and needs a very short period of time in order to get updated to the current instant. This assumption is quite realistic because most of the HA solutions available in market [61] employ the technique of buffering the service state for some time. So, in case of failover of a VM, the time for cold standby replica to come into service is very small, this time is indicated by restart intensity **μ** in Figure 16. The restarting of VM, that is booting from storage and updating, is covered by the factor **c**.

It can be noticed from Figure 16 that there are two possible states for the standby replica. It is either updated, marked by **U**, or lies in the storage, unpowered and not updated, marked by **S**. thus, as long as there is an updated standby, the service doesn't encounter any downtime even if the active replica fails [35]. But in case there is no updated standby running and a failure in active replica occurs, a downtime is introduced in the service as it waits for the replica to restart.

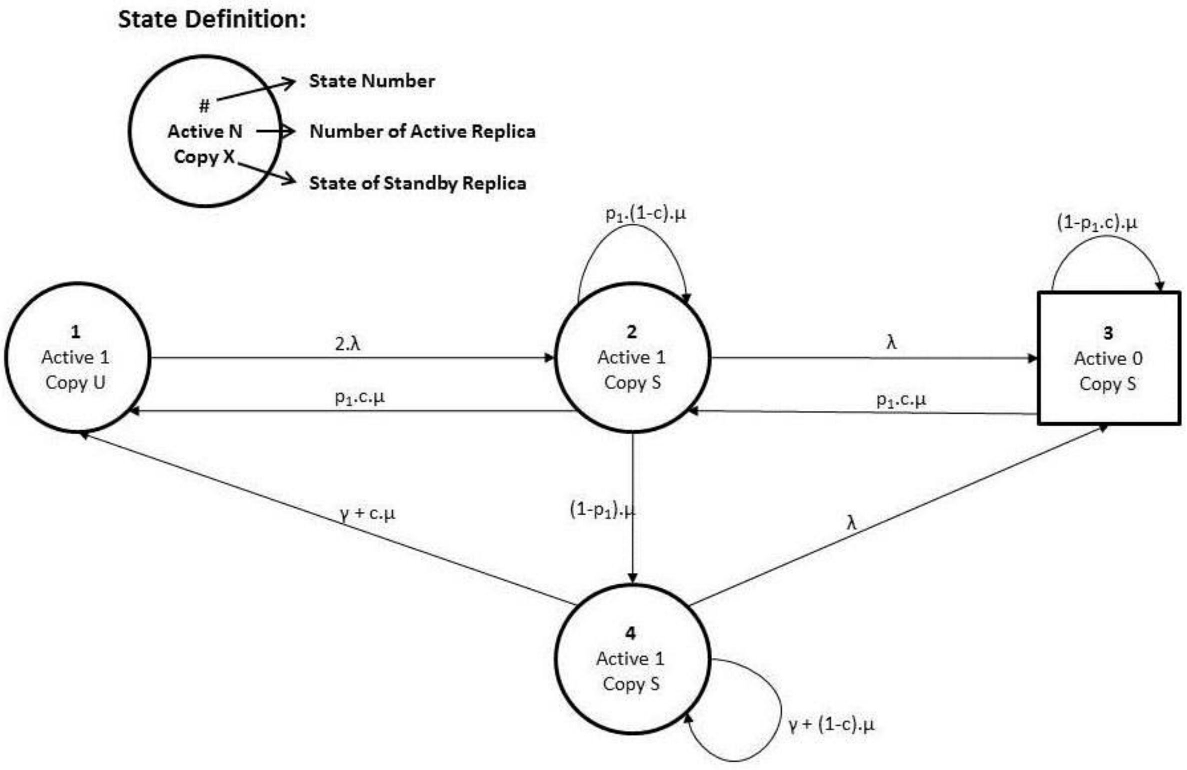

Figure 16: Markov Model of Updated Dedicated Hot Standby

It has been discovered in the literature that almost all HA solutions in the market run the two updated replicas in the same cluster [35], although some examples of having the two replicas running in different clusters are also found [61]. In the current work, it is assumed that both replicas are running in the same cluster. This will not only keep our models intact but also is practical as already stated.

Another consideration in the model is the load in the cluster. As a server fails, the resources in the cluster are reduced and in order for the management system to restart another replica, there have to be enough resources in the cluster. In Figure 16, $\mathbf{p_1}$ is the probability that there are enough resources available in the cluster for the replica to start. The management system is considered to assume the job of maintaining a certain level of working servers in the cluster. Thus, cluster boundaries are considered to be variable. In Figure 16, $\frac{\mathbf{1}}{\boldsymbol{\gamma}}$ is the time required for the management system to fix the cluster boundaries.

In order to make all the techniques comparable, the VMs are assumed to be similar to each other. This means that they have the same failure and restart intensities, coverage factors and are running in the same cluster implying that the time taken to fix cluster boundaries is also the same. It is only the service behavior and the probability of having enough resources for restarting a VM that are different for different techniques.

In order to simplify the model, different states in the model are examined and it is observed that **state 4** can be argued as very unlikely to be acquired by the service. The reason of this argument is that there is very shear probability that there aren't enough resources available in the cluster for starting a new VM. This argument is powered by the cloud datacenter utilization numbers obtained from the literature. Hence, it is observed that normal utilization of a datacenter ranges from 10-50% [34], although there are some spikes in the utilization. A test conducted by Intel revealed that the server utilization in virtualized datacenter is in the range of 35-42% with a maximum value of 56% [69]. In another case, however, the datacenter utilization is found to be 70% [70]. Nonetheless, the cluster utilization grouped with the fact that the service running this technique is always given a priority yields the assumption that the probability of having enough resources for starting a VM is one, i.e. $\boldsymbol{p_1 = 1}$. This leads to a simplified model as shown in Figure 17.

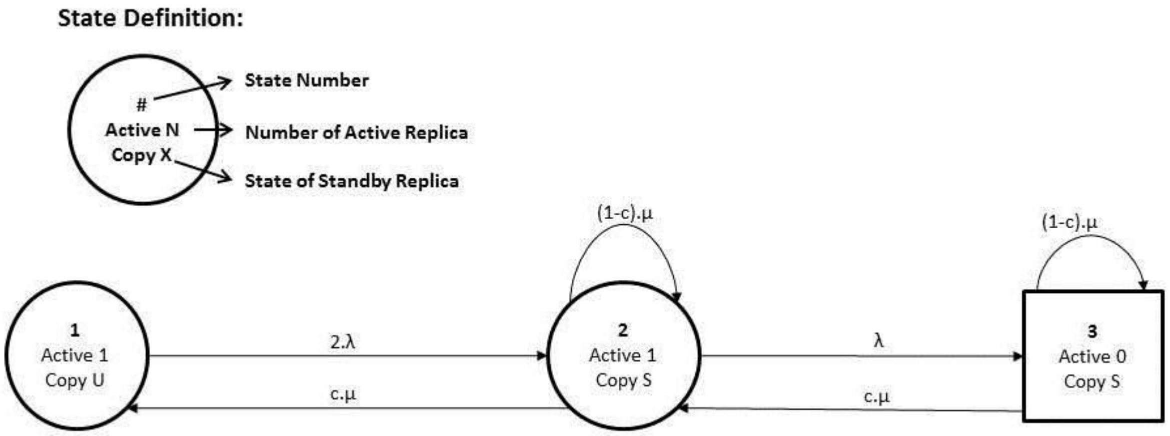

Figure 17: Simplified Markov Model of Updated Dedicated Hot Standby

The simplification has led the model to omit the service availability's dependence upon two parameters namely $\boldsymbol{\gamma}$ and $\mathbf{p_1}$. In order to validate this simplification, a numerical analysis is conducted on the above models and the results are compared. The numerical values for different parameters are enumerated in Table 3. For the original model, a plot of availability $\boldsymbol{A_{o,1}}$ against $\mathbf{p_1}$ is drawn, while for the simplified model, a single value of availability $\boldsymbol{A_{s,1}}$ is obtained. It is then observed from Figure 18 that the value ($\boldsymbol{A_{s,1} = 0.999971}$) is in the vicinity of values for $\boldsymbol{A_{o,1}}$ in the range of ($\boldsymbol{p_1 = 1.0}$) which implies that the simplified model can rightly be used to approximate the original model. It can also be deduced Figure 18 that even if there is 10% probability that the resources are not available for a starting a VM, the availability remain **0.999967**, which is very close to $\boldsymbol{A_{s,1}}$. This gives us even more confidence in using the simplified model because this observation shows that the simplified model can also cater for some blocking probability even though it is considered to be zero while coming up with the model.

Moreover, the plot gives an insight in the availability trend of this type of service. It is seen that even for **65%** of blocking probability in the cluster, **99.99%** availability is guaranteed for this type of service. This is mainly due to the reason that this service type is given priority over all other services and hence whenever there is competition for resources, it claims them.

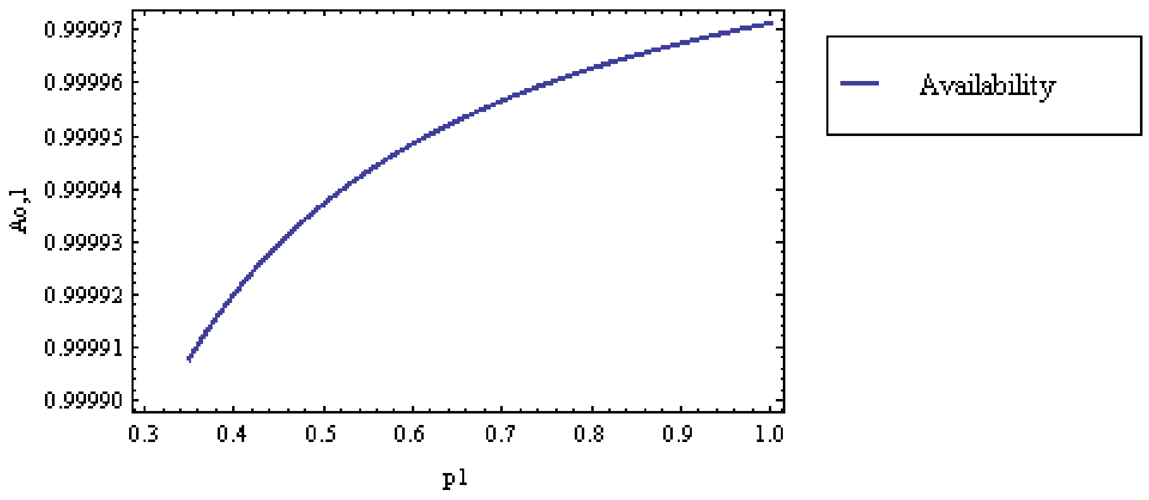

Figure 18: $A_{o,1}$ Versus p1 Plot for Validation of Simplification of the Model

5.2.2 Updated Shared Hot Standby

As already highlighted in Section 3.2, in this technique, there are two replicas of a service running. One of them is providing the output while the other is just getting the state updates from the active replica. In case of failure, the standby replica needs some time to get updated and hence introduces a brief downtime. In the model developed for this technique, Figure 19, this downtime is equal to$\frac{1}{\alpha}$.

In [36], the value mentioned for sending updates to the standby replica is 25 milliseconds which suggest that the value of downtime is in order of seconds. In this study, we make a careful selection by choosing one minute as the time required for a standby replica to get updated and start producing output. Although it is relatively higher value compared to the anticipated value, we use it so that the results can hold even for slower updating schemes. Also the assumption that both hot replicas are running in the same cluster is hold in this case also for the same reason as for the previous case.

All the other parameters in Figure 19 are the same as for the model drawn in Figure 16 except for the probability that there are enough resources to start a new VM following this technique. In this case, it is denoted by $\mathbf{p_2}$ and it is different from $\mathbf{p_1}$ in the sense that it means that there are enough resources for starting a new VM of this type even after starting all the required VMs for hosting updated dedicated hot standby replicas. This imposes extra constraints on the availability of resources for this type compared to the previous case.

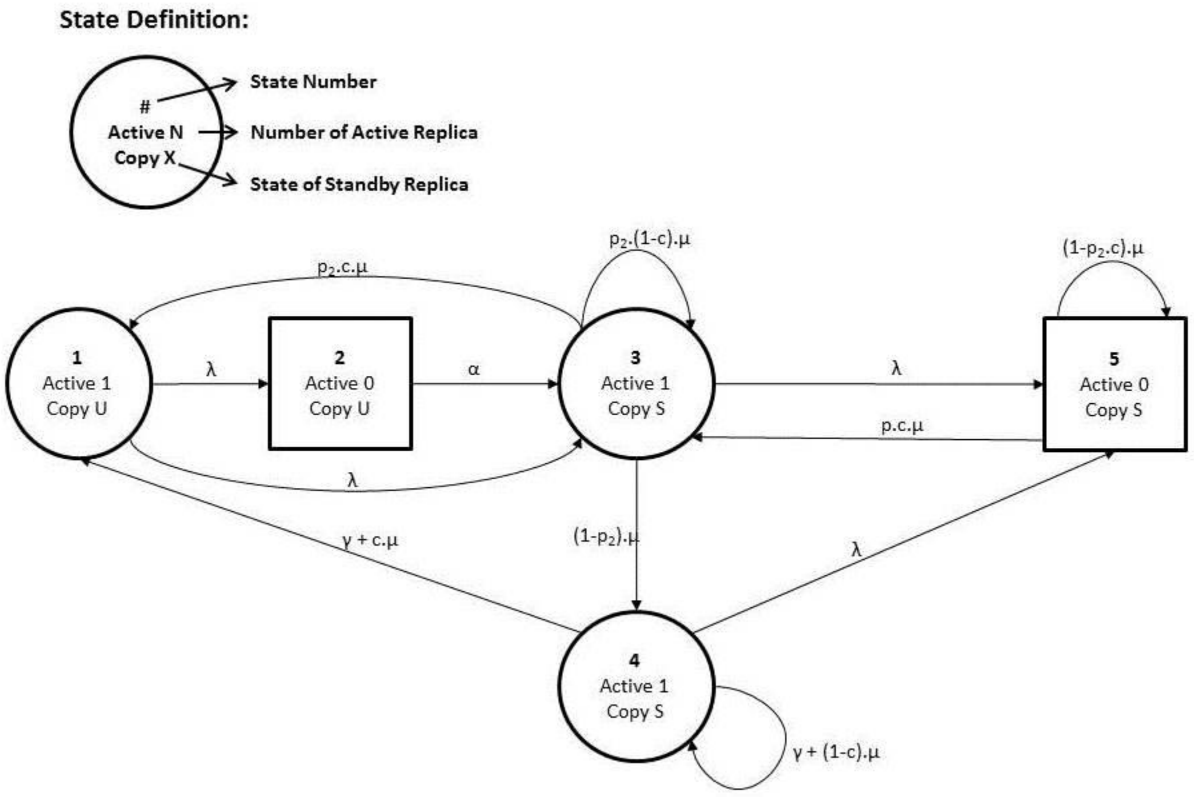

Figure 19: Markov Model for Updated Shared Hot Standby

In this case also, the model is simplified by assuming that there are always enough resources for starting a new VM for a service whenever it is needed. This assumption holds due to the fact that datacenter utilization has, under normal circumstances, found to be less than 50% which clearly indicates that there are enough resources to start a new replica of each running service in the datacenter. Therefore, **state 4** in Figure 19 is eliminated. The effects are obvious and that include; fewer number of parameters on which service availability expression depends and fewer states of the system making it easy to solve. The resulting model is shown in Figure 20.

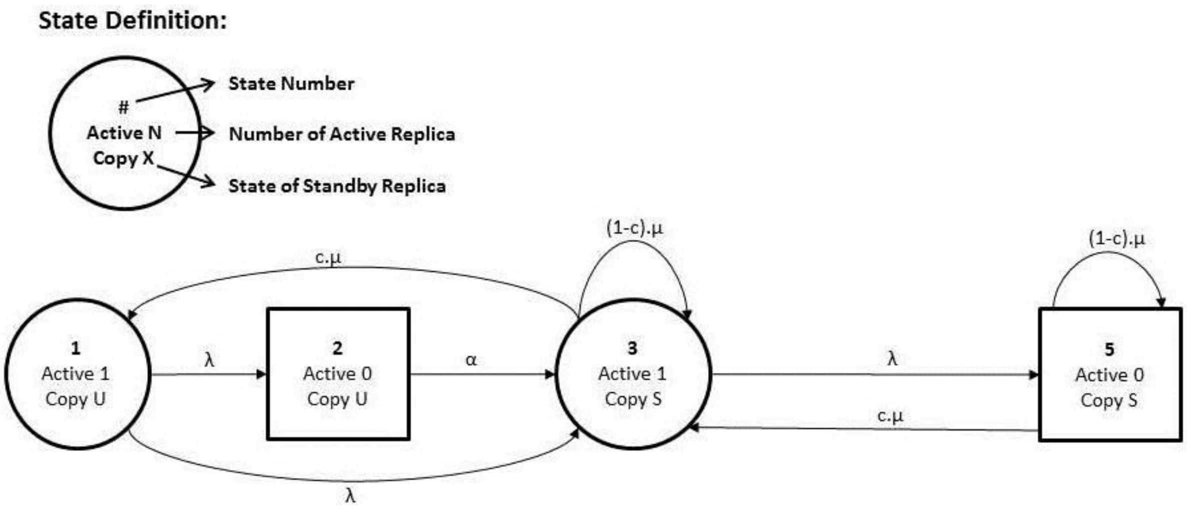

Figure 20: Parsimonious Markov Model for Updated Shared Hot Standby

The same analysis is carried out for validating this simplification as previously done for updated dedicated hot standby technique. In the current scenario, the availability of service $\boldsymbol{A_{o,2}}$ versus probability of available resources to start a new VM $\mathbf{p_2}$ is plotted as shown in Figure 21 by solving the original model as depicted in Figure 19. Also a value for availability $\boldsymbol{A_{s,2}}$ is calculated by solving the simplified model given in Figure 20.

As a result, the availability value ($\boldsymbol{A_{s,2} = 0.999852}$) of the simplified model is found to be the same as that of the original model for ($\boldsymbol{p_2 = 1}$). In addition it is also observed from the plot in Figure 21 that the value of service availability is ($\boldsymbol{A_{o,2} = 0.999848}$) even with 10% probability that there are not enough resources for starting a new VM for the service using this dependability technique. This value is still very close to the availability value of the simplified model. Therefore, this gives a hint that the simplification is not only good enough for representing the case of no blocking at all, but can also account for cases with small probabilities of blocking in restart of new VMs.

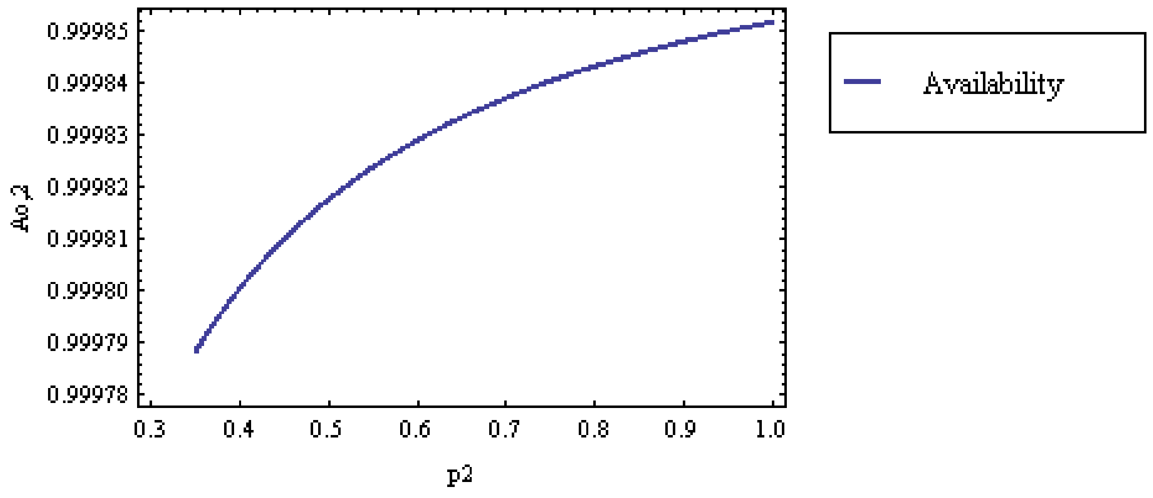

Figure 21: $A_{o,2}$ Versus p2 Plot for Validation of Simplification of the Model

The trend observed in Figure 21 is the same as that for updated dedicated hot standby and hence same comments apply for this trend also. Therefore, this technique also doesn't distort the availability very much, rather keeps it within the limit of **99.97%**.

5.2.3 Cold Shared Standby (High Priority Service)

Unlike the previous two schemes, this one has only one active replica and the other replica is lying unpowered in the storage of cloud service provider. In order to keep the cold standby on the same level of availability and responsibility as the cold standbys of the previous schemes, its RTO and RPO are considered to be the same as for the previous techniques.

One additional remark about this technique is that there are two different techniques proposed with shared cold standbys. There is no difference in the state of the standby replica in them. Instead, the difference lies in the priority. Hence the higher priority service has an opportunity to preempt the VM running a lower priority service to give way to start its own VM. In this way, higher priority service will have higher availability than the lower priority service when the datacenter is overloaded.

The probability that there are enough resources for restarting VMs employing this technique is again dependent upon the cluster load and number of VMs to be restarted with higher priority level. Hence the probability is indicated in the model of Figure 22 as the parameter $\mathbf{p_3}$.

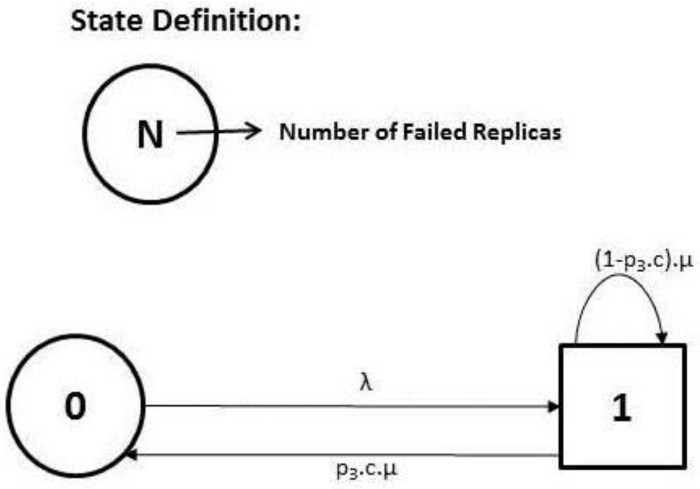

Figure 22: Markov Model for Cold Shared Standby (High Priority Service)

The same justifications for simplification that are used for the previous two schemes are applicable for the model shown in Figure 22 also. Thus by performing the simplification, it is noticed that service availability becomes independent of $\mathbf{p_3}$, this is seen in Figure 23.

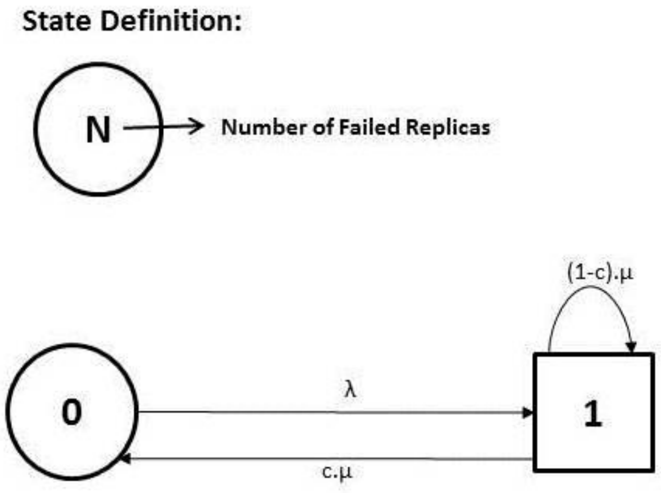

Figure 23: Simplified Markov Model for Cold Shared Standby (High Priority Service)

In order to validate the simplification assumption, the same approach as for the previous schemes is employed. Hence a plot of availability $\boldsymbol{A_{o,3}}$ of the original model sketched in Figure 22 against $\mathbf{p_3}$ is drawn, as shown in Figure 24. Also, the availability value $\boldsymbol{A_{s,3}}$ of the simplified model of Figure 23 is calculated and is found to be ($\boldsymbol{A_{s,3} = 0.996214}$).

It is clearly seen from the plot in Figure 24 that the values for $\boldsymbol{A_{o,3}}$ range from **0.9958** to **0.9962** for the respective range of $\mathbf{p_3}$ values from **0.9** to **1.0**. These values lie in very close vicinity of $\boldsymbol{A_{s,3}}$**.** This observation justifies our simplification and also suggests that the simplification doesn't only account for the case when there are enough available resources for starting a new VM, but also caters for the case when there is as much as 10% probability of not having enough resources for the purpose. Another observation made in this regard is that the availability of service employing this technique decays rapidly with increasing cluster load. This observation is justifiable because higher cluster load and lower priority of this service type makes it unlikely for it to claim resources readily. Also, the fact that it doesn't have any hot standby, which will take over in case of failure, increases its downtime dramatically. Now, its availability solely depends upon how soon it will get resources to restart the VM and resume service provision.

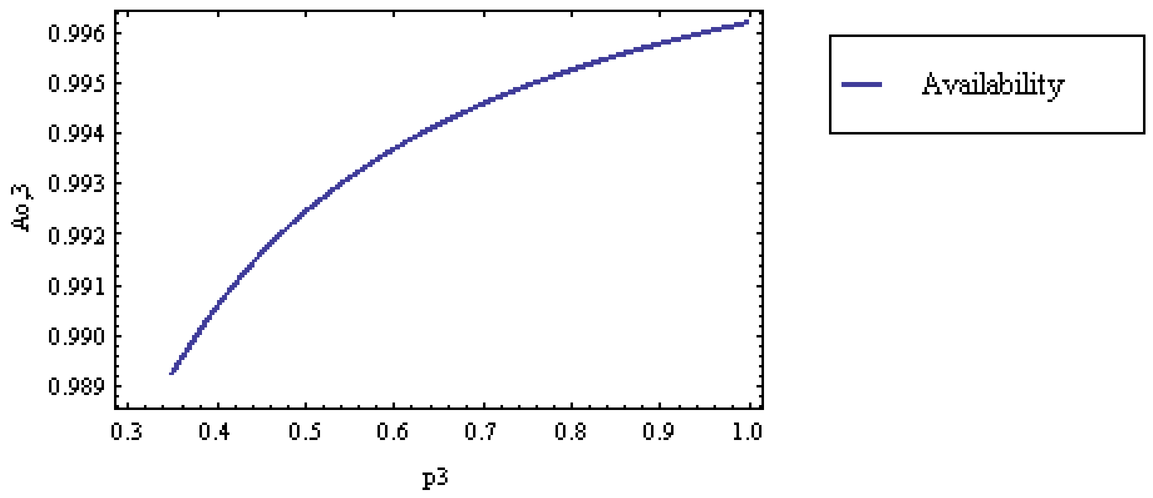

Figure 24: $A_{o,3}$ Versus p3 Plot for Validation of Simplification of the Model

5.2.4 Cold Shared Standby (Low Priority Service)

This scenario is the same as the previous one in all its aspects. Therefore, all the assumptions stated for the previous scenario also hold for this one. One additional feature of this technique is, though, that it can be preempted by the services employing the previous dependability techniques. This makes the services running this technique to have lowest priority among the services hosted by a cloud provider.

This deviation from the previous model has caused addition of a system state. Thus, the state **Q** represents the state of the system when there aren't enough resources for the service to start a new VM. Moreover, the ability of higher priority services to preempt the service running this technique introduces a new transition parameter called preemption intensity **δ.** It is clear from the description of this parameter that it is a function of the cluster utilization. But to keep our model simple, this dependence is not exploited in analytical models, but has been accounted for thoroughly in simulation study, and the value for preemption rate is taken to be constant at **0.2**, Table 3. Thus, the Markov model for this technique is drawn and is represented in Figure 25.

The availability of service employing this technique is also checked for sensitivity against $\mathbf{p_4}$. This is done by plotting $A_{o,4}$ against $\mathbf{p_4}$ as done in Figure 26. It is interesting to notice that unlike all the other techniques, which have logarithmic curves, the plot for this technique is rather linear. This has the implication that the decay from **1.0** towards **0.0** is rapid. This is one reason why it is deduced that this model can't be simplified. The other reason in this regard is the fact that for clusters having higher than 50 % utilization, the chances of the service being preempted is pretty high, depending upon the rapidness of failure occurrences and number of higher priority services.

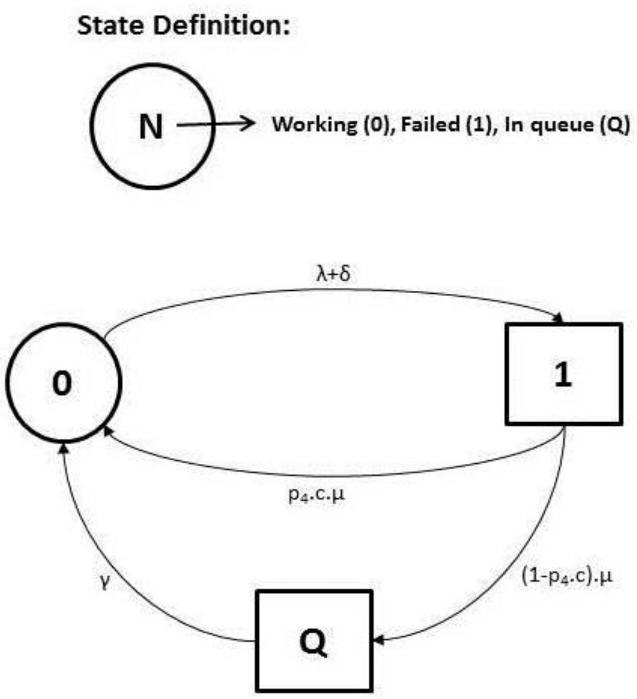

Figure 25: Markov Model for Cold Shared Standby (Low Priority Service)

This behavior is quite vivid in Figure 26 where the availability values for heavily loaded clusters are very low. For preempting intensity of one every ten hours and moderately loaded cluster with probability of not having enough resources for restarting a VM equal to 10% , the availability is in the order of **0.9**.

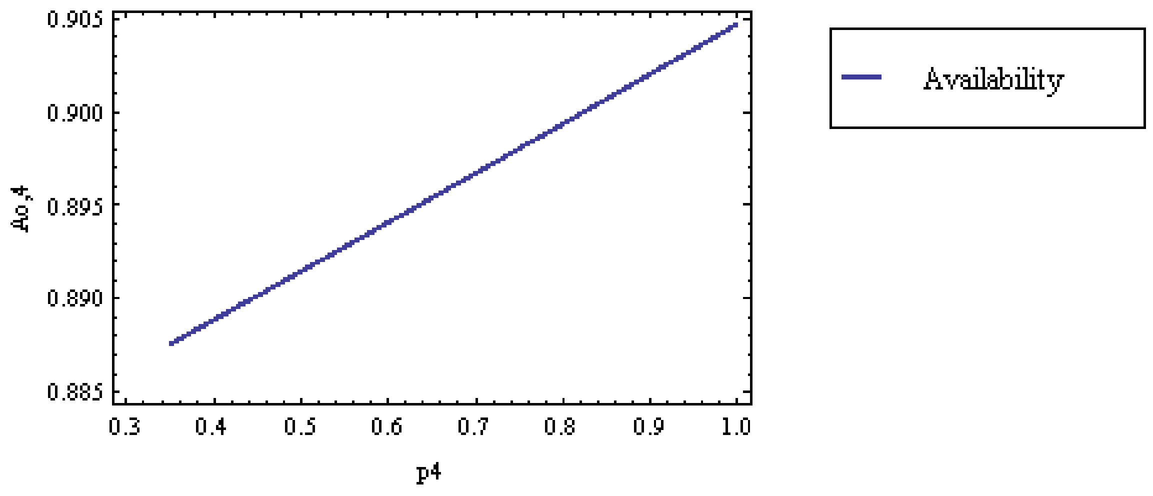

Figure 26: $A_{o,4}$ Versus p4 Plot for Validation of Simplification of the Model

In the above, all VM level techniques used for differentiating dependability of services have been described and their models for analysis have been drawn. Initial analysis of the models has also been performed. Thus, the models are simplified and the simplification is validated using numerical values from Table 3.

5.3 Cloud Level Models

In this section, reliability block diagrams for different scenarios of spatial separation of standby replicas are developed, solved and simplified for finding the service availability of cloud service. There are a large number of scenarios which can be studied, but in this work only four scenarios are considered. Before listing down the scenarios, it is worth-mentioning that VM level models have direct implications on the cloud level models. The reason is that depending upon how many replicas are needed; their spatial separations can be altered for different scenarios.

The approach taken in developing these models is that the underlying physical structure of the cloud, discussed in Chapter 4, is considered and each physical entity is modeled as an entity in the block diagram. This makes the service availability directly computable knowing the availability values for individual entities. It is worth-mentioning that the availability value of each entity actually encompasses all the possible failures in the entity.

Therefore, the four different scenarios selected for studying dependability differentiation depending upon spatial separation of replicas are:

1. All replicas in the same cluster of a public cloud
2. All replicas in the same datacenter of a public cloud
3. All replicas in the same public cloud (different datacenters)
4. Replicas in a hybrid of a public and a private cloud

The selection of these scenarios is not random. They are carefully selected in order to incorporate as much of the diversity as possible without compromising on their realism and avoiding as much of overlapping of conditions as can be avoided. This means that for most of the cases, examples can be found in the literature; for instance scenarios one and two are found in [61]. This also means that the scenarios are widely different from one another. And finally, this means that the minimum and maximum spatial separation which is reasonable to be provided to the services is included along with the main step-sizes between them.

An observation about scope of cloud management systems is also worth mentioning at this point. It is seen in Figure 4 that cloud management systems can be divided into three basic layers; namely VM management layer, VI management layer and Cloud management layer. This observation, combined with the physical architecture of our system under study from Chapter 4, is used to decide the scope of each management layer. Some of these scopes are also justified in the literature and the others are deduced by analogy. Hence the scope of VM layer is a cluster [61] [62], the scope of VI layer is a datacenter and the scope of cloud layer is a cloud. An additional layer of management is also used in the current study, called the service management layer, in order to cater for services running in a hybrid cloud. This management layer is already introduced and described in Chapter 4.

To start with, let us deal with the power distribution and cooling system. It can be seen from the structural model of the system that both power and cooling system are required to be available for guaranteeing the availability of the service. A series block diagram is thus proposed. This diagram is shown in Figure 27 and it can be seen from the figure that all physical entities required for powering a datacenter and cooling it are considered.

A method for aggregation of site attributes (power distribution of datacenter and cooling system) is used which is provided in [64]. According to this method, instead of calculating the availability of different systems by normal methods of finding availability of systems in series, the minimum of all is taken. Hence in our system the power structure for datacenter and the cooling system are aggregated into a single entity called site attributes as shown in Figure 27.

By looking into the power distribution scheme of the system under study, it is found to fit in the description of Tier II system which is illustrated in [64]. Hence all of the entities in the model are considered to be Tier II and hence the availability value for the entity **Site Attributes** is fixed to be **0.9975** [64] and used throughout this study, Table 4.

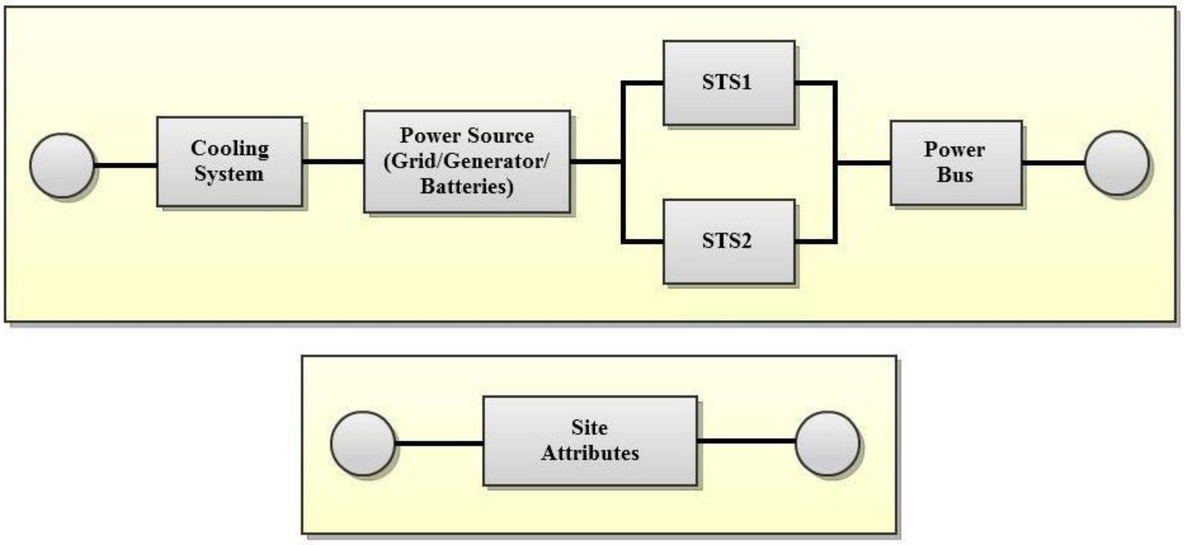

Figure 27: Aggregation of Site Attributes

In the following, the model for each scenario is drawn, solved and simplified in order to come up with parsimonious model, and analyzed for service availability by using different VM level dependability techniques. The models were first introduced in the preceding project work [12]. In this work, they are tuned for analyzing wider range of dependability techniques.

All block diagrams are solved by using packages developed in Mathematica 6.0 by B. E. Helvik [68] for solving block diagrams. The availability expression for each model follows it. This expression is analyzed for each model to find parameters that can be dropped in order to simplify the model. Then the simplified model is presented along with its expression. This model is then validated using numerical values found in literature and tabulated in Table 4.

5.3.1 Scenario I: All Replicas in the same Cluster of a Public Cloud

In this scenario, it is assumed that all the replicas, hot and cold, reside in the same cluster. Implicitly, it means that the same cloud provider manages the whole provision of the service. The structural model for service provision developed in Section 4.2.4 and visualized in Figure 13 till Figure 15 is used to come up with the block diagram of Figure 28 representing this scenario. In the figure, the entity **Server Matrix** is used for referring to the array of servers which host different replicas depending upon the VM level techniques implemented. Thus availability values for different such techniques are put here for the purpose of analysis.

A brief account of how management system is dealt with will be instrumental at this stage. As already pointed out, there are four layers of management in total. In the block diagrams developed henceforth, all management layers in series to each other are aggregated together. This is shown by mentioning the layers aggregated in the **Management System** entity in the diagrams. The availability values for all the management layers are considered to be equal.

So, the subscript $\boldsymbol{x}$ in $\boldsymbol{A_{mngt,x}}$ appearing in the availability expressions hold for the number of layers aggregated together and nothing else.

For simplifying the model in Figure 28, the availability expression is examined and it is observed that the terms $\boldsymbol{(1 - A_{pdu})^2}$ can actually be dropped as it represents very small quantity, Table 4. This reduces the model to Figure 29 and the availability expression to $\boldsymbol{A_{scenario\ I,simple}}$. The values of availability for both the cases are calculated for updated hot standby and it is clear that the simplification doesn't affect the final result. Hence the simplification is justified.

One more observation made is that although the term $\boldsymbol{1 - (1 - A_{switch})^2}$ looks promising for dropping as it has high order values of very small number, but the fact is that the value of availability of switch is pretty low, Table 4. This causes the term mentioned to remain effective in the availability expression and unjustifiable for neglecting.

5.3.2 Scenario II: All Replicas in the same Datacenter of a Public Cloud

In this scenario, different standbys are handled differently. Hence the hot standbys are hosted in the same cluster as the active replica but the cold standbys are residing elsewhere in the datacenter. Hypothetically, as the independence among the failures of active and cold standby is increased, the availability is argued to be increased as well. To study this availability, a block diagram of the system is drawn as shown in Figure 30.

In the Figure 30, the entity **Server Matrix** has now a smaller scope than it has for the previous scenario. Now, it only shows the servers running the hot replicas of a service. Thus, in case a service has both an active replica and a hot standby, both the servers are covered by **Server Matrix.** But in case the service doesn't have any hot standby, then **Server Matrix** entity only represents the server running the active replica.

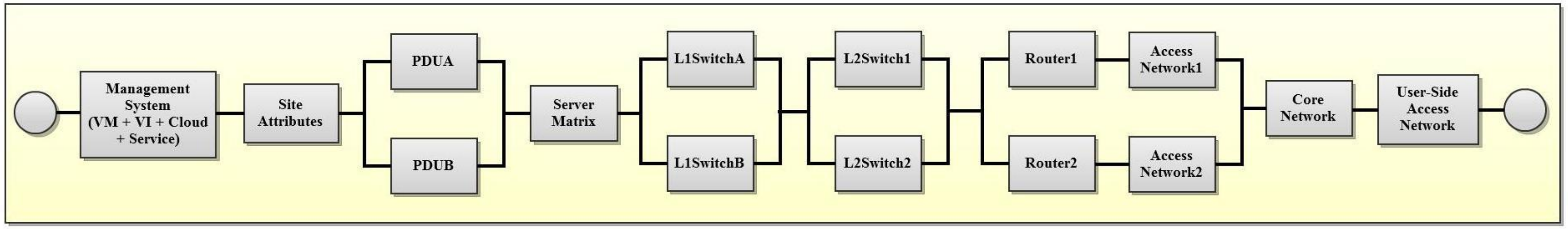

Figure 28: Scenario I: All Replicas in the same Cluster of a Public Cloud

$$A_{scenario\ I} = A_{user}.A_{core}.A_{mngt,4}.\left(1-\left(1-A_{pdu}\right)^2\right).(1-(1-A_{access}.A_{router})^2).A_{server}.A_{site}\ (1-(1-A_{switch})^2)^2$$

$$A_{scenario\ I} = 0.981645$$

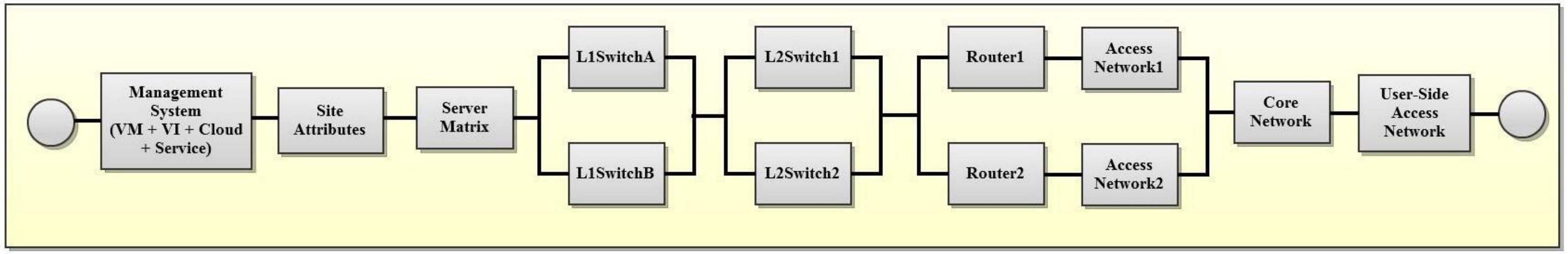

Figure 29: Simplified Model of Scenario I

$$A_{scenario\ I,simple} = A_{user}.A_{core}.A_{mngt,4}.(1-(1-A_{access}.A_{router})^2).A_{server}.A_{site}\ (1-(1-A_{switch})^2)^2$$

$$A_{scenario\ I,simple} = 0.981646$$

The entity used to refer to the storage in which the cold standby lies is denoted as **Standby Storage** in Figure 30. In this study, the availability value of **Standby Storage** is taken to be **1.0**. This assumption is inherited from the VM level modeling in which the failure of cold standby is not considered. This assumption is strengthened by the fact that as the replica is not powered, it is very less exposed to faults leading to failures as there is no wear and tear, no executions which can cause logical errors and no monitoring and management which can cause operational failure [5]. But in case of VM level models, the possibility of the cold standby failing to start as required is covered by a factor which is missing in the block diagram due to its limitations.

One more observation made regarding this model is that the management system entity has already been dismantled. Hence the there are two management entities now, one specific for each cluster and the other for the rest of the cloud system.

It is observed from the availability expression that the same term which is dropped in the previous scenario can be dropped here also for the same reasons. After the simplification is implemented, a new simplified model is obtained which is shown in Figure 31. It is clear from the figure that PDUs are eliminated from the model as their contribution to the final availability value is negligible. This is verified by calculating the availability values for both the models considering the service running to have a hot updated dedicated standby.

Also the entity **Standby Storage** is removed from the model. The reason is that it is assumed that the value of parameter $\mathbf{A}_{\mathbf{storage}}$ is equals to **1.0**. The reason for this assumption has already preceded.

Therefore, a model with two less parameters than the original model is resulted after the simplification is done.

5.3.3 Scenario III: All Replicas in the same Public Cloud

In this scenario, the replicas are separated to decrease the number of single points of failures even more. This is done by storing the cold replica in a totally different geographical location (i.e. datacenter) from where the active replica(s) are running. The resulting block diagram is given in Figure 32.

The simplification is also done by dropping the same term as in the previous two cases as there isn't any other term by dropping which an almost unaltered final result is obtained. The simplified model is given in Figure 33 and is followed by the availability expression and calculated value of availability for updated hot standby. At this stage, only values for hot updated standby are considered for the validation of the simplifications performed. The detailed analysis for different techniques will follow later.

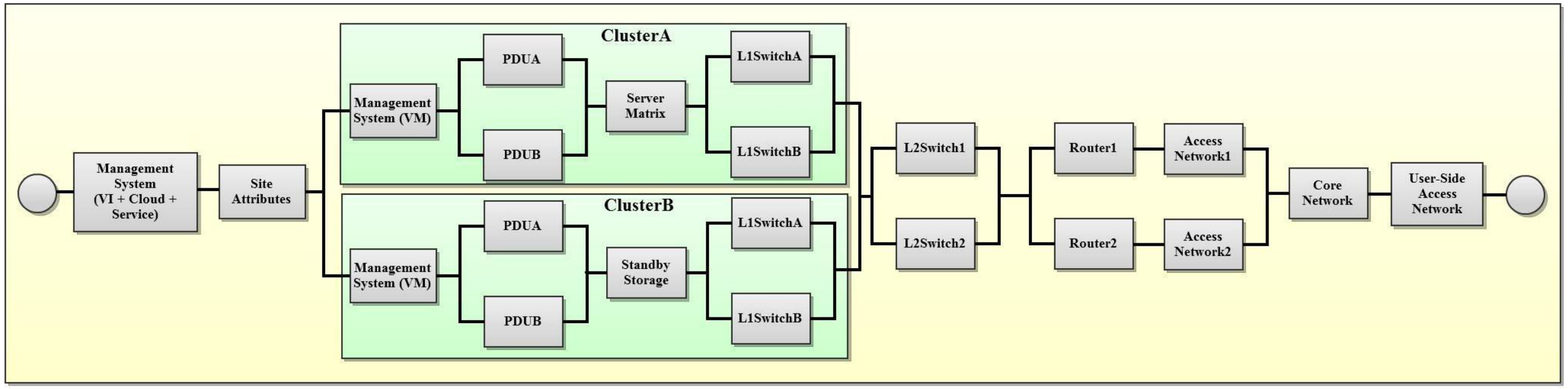

Figure 30: Scenario II: All Replicas in the same Datacenter of a Public Cloud

$$A_{scenarioII} = A_{user}.A_{core}.A_{mngt,3}.A_{site}.(1-(1-A_{access}.A_{router})^2)(1-(1-A_{switch})^2)$$

$$\left(1-\left(1-\mathrm{A}_{mngt,1}.\left(1-\left(1-\mathrm{A}_{\mathrm{pdu}}\right)^2\right).\mathrm{A}_{\mathrm{server}}.(1-(1-A_{switch})^2)\right)\left(1-\mathrm{A}_{mngt,1}.\left(1-\left(1-\mathrm{A}_{\mathrm{pdu}}\right)^2\right).\mathrm{A}_{\mathrm{storage}}.(1-(1-A_{switch})^2)\right)\right)$$

$$A_{scenario\ II} = 0.983054$$

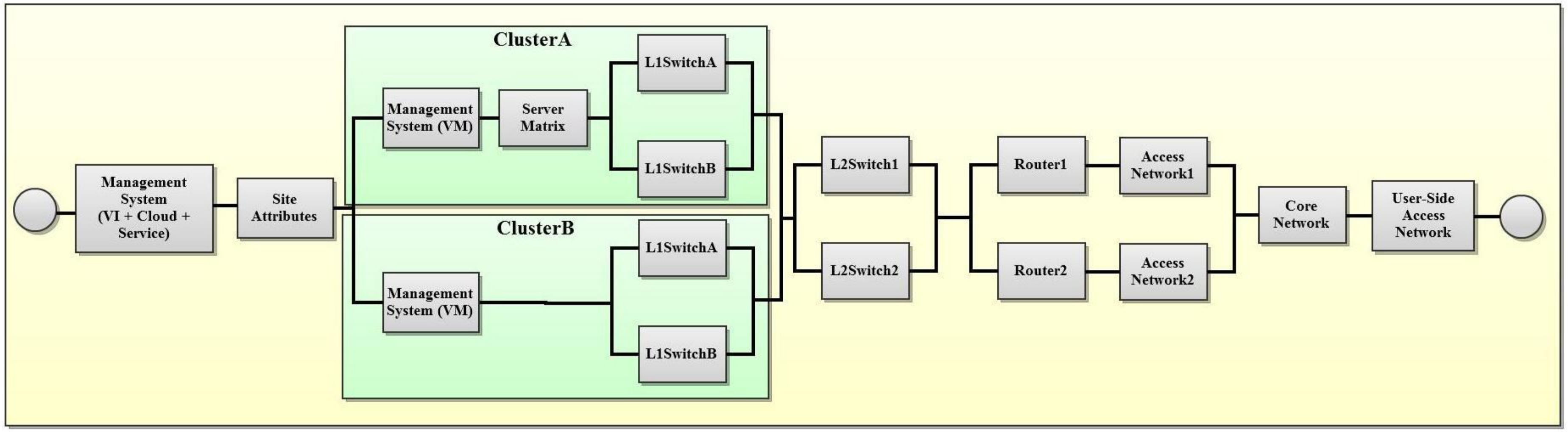

Figure 31: Simplified Block Diagram for Scenario II

$$A_{scenarioII,simple} = A_{user}.A_{core}.A_{mngt,3}.A_{site}.(1-(1-A_{access}\ .A_{router})^2)(1-(1-A_{switch})^2)$$
$$\left(1-\left(1-\mathrm{A}_{mngt,1}.\mathrm{A}_{\mathrm{server}}.(1-(1-A_{switch})^2)\right)\left(1-\mathrm{A}_{mngt,1}.(1-(1-A_{switch})^2)\right)\right)$$

$$A_{scenario\ II,simple} = 0.983054$$

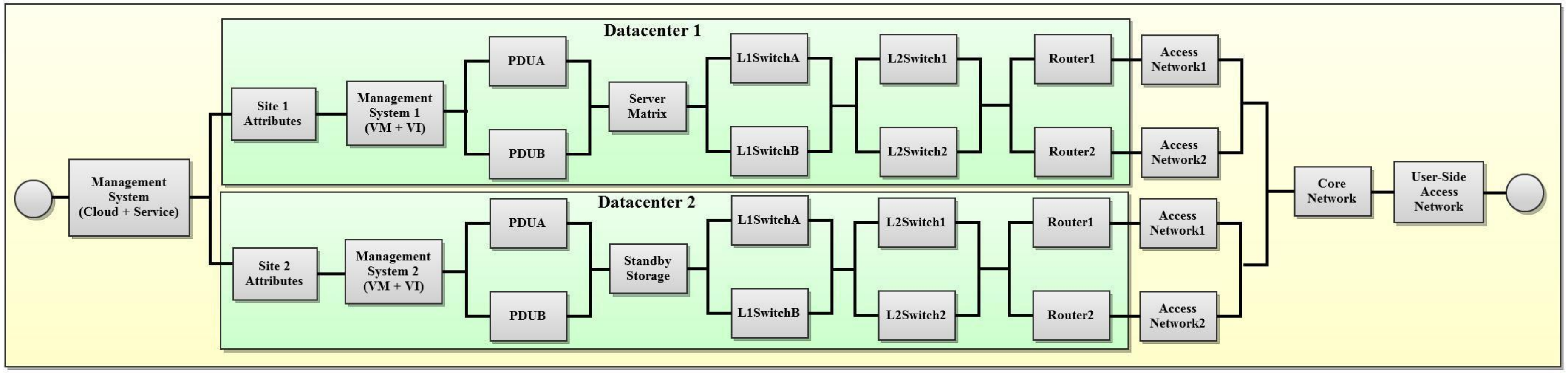

Figure 32: Scenario III: All Replicas in the same Public Cloud

$$A_{scenarioIII} = A_{user}.A_{core}.A_{mngt,2}.$$

$$\left(\begin{array}{c} 1-\left(1-A_{mngt,2}.\left(1-\left(1-A_{pdu}\right)^{2}\right).A_{server}.A_{site}.(1-(1-A_{switch})^{2})^{2}(1-(1-A_{access}.A_{router})^{2})\right). \\ \left(1-A_{mngt,2}.\left(1-\left(1-A_{pdu}\right)^{2}\right).A_{storage}.A_{site}.(1-(1-A_{switch})^{2})^{2}(1-(1-A_{access}.A_{router})^{2})\right) \end{array}\right)$$

$$A_{scenario\ III} = 0.987004$$

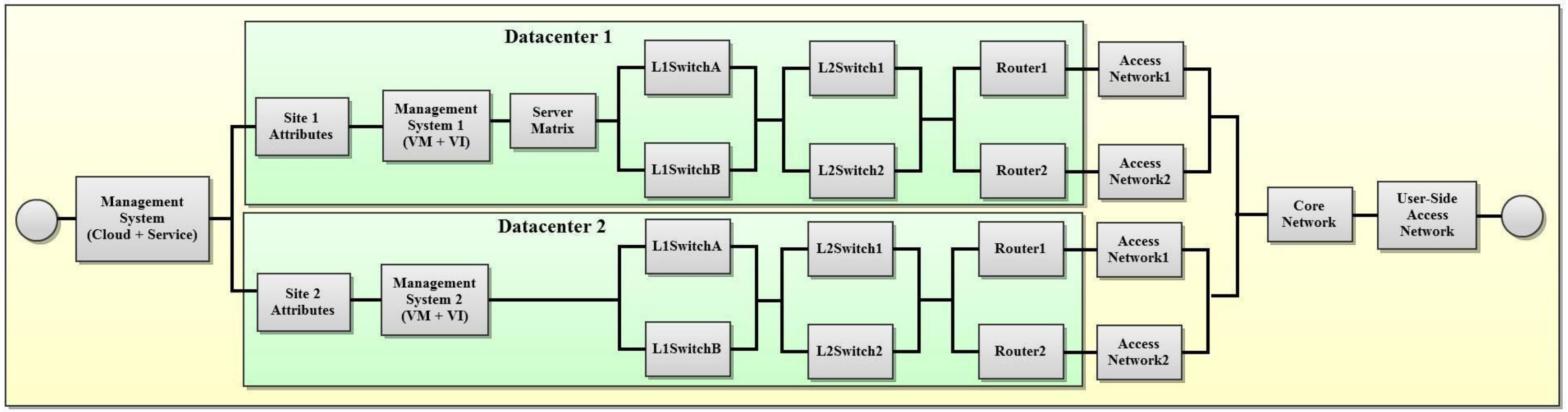

Figure 33: Simplified Model for Scenario III

$$A_{scenarioIII,simple} = A_{user}.A_{core}.A_{mngt,2}.\begin{pmatrix} 1-\left(1-A_{mngt,2}.A_{server}.A_{site}.(1-(1-A_{switch})^2)^2(1-(1-A_{access}.A_{router})^2)\right). \\ \left(1-A_{mngt,2}.A_{site}.(1-(1-A_{switch})^2)^2(1-(1-A_{access}.A_{router})^2)\right) \end{pmatrix}$$

$$A_{scenario\ III,simple} = 0.987004$$

5.3.4 Scenario IV: Replicas in Hybrid Cloud

This scenario differs from the previous scenarios in the sense that in this case a combination of public clouds making a hybrid is considered. In light of above given outline, a block diagram for this scenario is drawn in Figure 34. It is worth noticing that the service management layer remains the only single point of failure, along with the core and user access networks. As already stated, this layer can be provisioned by a third-party provider or can be managed by the service user himself.

The failure and recovery occurs in this scenario following the forth-mentioned pattern. In case the primary service providing cloud fails, then the secondary service provider will become the primary provider and the service management will either try to make the previous primary as secondary, in case it is available, or will look for another secondary cloud service provider.

As this scenario is very closely related to scenario III, the same simplification measures are taken in this case also. The resulting model is given in Figure 35 and the availability expression follows it immediately. Like all the previous scenarios, the simplification is validated and found to be valid as the resulting availability values for the original model and the simplified model are exactly the same.

In this section, reliability block diagrams for analyzing different scenarios of differentiating dependability of services on cloud level have been drawn. Preliminary analysis of these models has also been performed in this section. This includes the simplification of models in order to come up with parsimonious models. The simplification has also been validated using numerical values from Table 4.

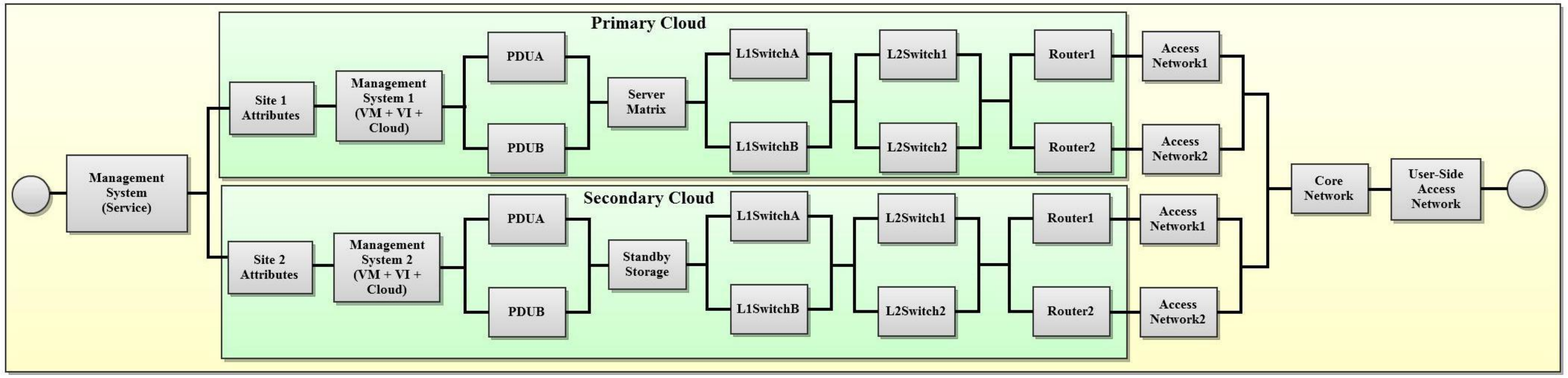

Figure 34: Scenario IV: Replicas in Hybrid Cloud

$$A_{scenarioIV} = A_{user}.A_{core}.A_{mngt,1}.\left(\begin{array}{c}1-\left(1-A_{mngt,3}.\left(1-\left(1-A_{pdu}\right)^{2}\right).A_{server}.A_{site}.(1-(1-A_{switch})^{2})^{2}(1-(1-A_{access}.A_{router})^{2})\right).\\ \left(1-A_{mngt,3}.\left(1-\left(1-A_{pdu}\right)^{2}\right).A_{storage}.A_{site}.(1-(1-A_{switch})^{2})^{2}(1-(1-A_{access}.A_{router})^{2})\right)\end{array}\right)$$

$$A_{scenario\ IV} = 0.98798$$

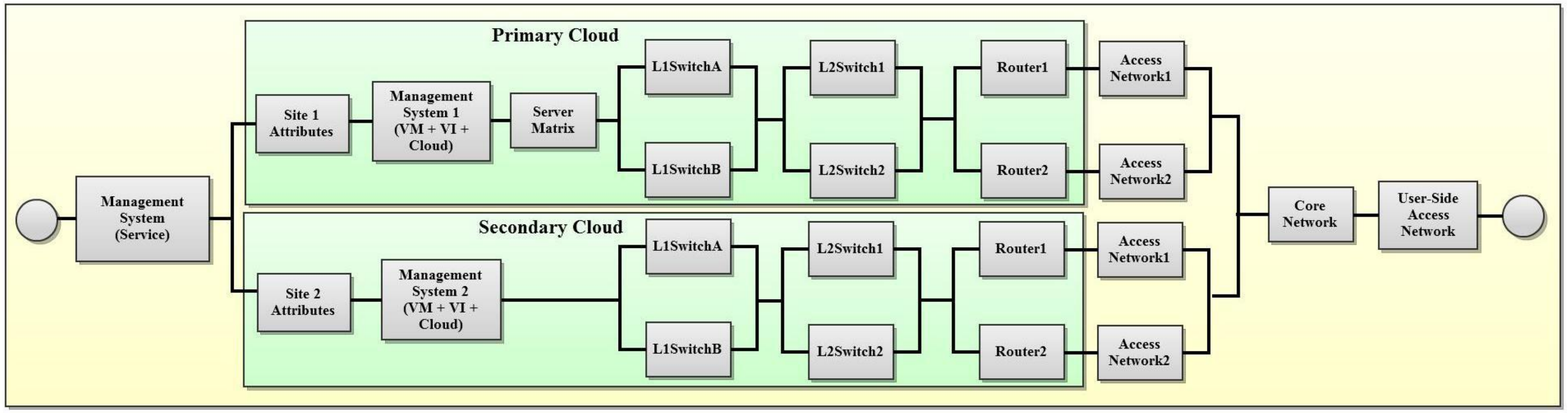

Figure 35: Simplified Model for Scenario IV

$$A_{scenarioIV,simple} = A_{user}.A_{core}.A_{mngt,1}.\begin{pmatrix} 1-\left(1-A_{mngt,3}.A_{server}.A_{site}.(1-(1-A_{switch})^2)^2(1-(1-A_{access}.A_{router})^2)\right). \\ \left(1-A_{mngt,3}.A_{site}.(1-(1-A_{switch})^2)^2(1-(1-A_{access}.A_{router})^2)\right) \end{pmatrix}$$

$$A_{scenario\ IV,simple} = 0.98798$$

6 Simulation – Modeling and Validation

In addition to the analytical models developed for conducting a study of dependability differentiation in cloud environment, a simulation based study is also performed. The need of simulator has arouse from the fact that the analytical models are developed by making a lot of assumptions, some of which can be relaxed in simulation. This chapter gives an introduction of simulation-based study and its advantages over analytical models. It also covers the design of simulator used in the present work, its models and finally validates the simulator developed against the analytical models.

6.1 Background

Simulation is a useful tool when it is not easy to obtain an analytical model or to solve it [71]. Simulations can also be used to evaluate accuracy of analytical solutions. A simulation is defined as, "the imitative representation of the functioning of one system by means of the functioning of another system" [55]. In most cases, a system is simulated by a use of a program, referred to as a simulator.

There are a number of advantages introduced by the use of simulations for system evaluation [55]. First of all, the analytical models are sometimes made too simple so that they can be analyzed mathematically. On the other hand simulations can handle complexity to some extent. Although they don't represent the real-world scenario completely, but still they give closer approximation of real-world conditions as compared to analytical models. Another advantage which simulations have is that they provide the exact level of granularity required by the system. One more feature of simulation, which makes analytical models easier than simulator development, is that the simulation development needs a deep insight into the system and its operations, so that all the necessary details about the system are included in the simulation. Although this leads to an insight into the system and helps in identifying the weaknesses it possesses but it costs more labour and time in return.

6.2 Choice of Simulation Model Type and Simulation Tool

There are different types of simulation models outlined in [55]. For the sake of this project, the attention is confined to **dynamic discrete event stochastic simulations**. A dynamic simulation is the one which shows the system as it changes over time. A discrete simulation is referred to the case when the variables in the simulator change only at discrete times (events). And a stochastic simulation is the one which includes some random variables causing some random output from the simulator.

The dynamic discrete event stochastic simulation is used in this project because the only points of interest are when the system components fail which are discrete events. Also, the randomness in the failure process of the system components is a well-known fact. By use of simulations, this randomness is easily modeled. One more advantage of this type of simulation over analytical modeling, specifically Markov model, is that the distribution of the random variables can readily be changed in simulation. As failures are rare-events and occur not very frequently, it is logical to use rare-event simulation with object-oriented simulation [72]. This may be a part of future work to be carried out in simulating cloud services and account for their dependability.

Although there exist a number of tools, in this project Simula programming language along with its context class Discrete Event Modeling On Simula (DEMOS) is used to simulate the failures in a server. The reason for using this platform is that Simula supports object-oriented programming and hence inherits all the pros of object oriented programming languages. Also it has a very powerful support for simulation modeling [73] as it was originally developed for simulation purposes. DEMOS context class makes it even more attractive for using because DEMOS provides functionality for simulating discrete events with minimum efforts of coding [74]. However some minor changes are made to the DEMOS class in order to customize it according to our requirements. Hence the edit function and report generation is altered in DEMOS.

6.3 Simulator Description

In the current study, a simulator is developed to simulate a cluster of virtualized servers in a cloud. In this study the number of servers is fixed to **1000** which classifies this cluster as a small sized cluster. The reason for keeping the size of cluster small is that by doing so, not only the simulation runs efficiently but also it meets the objective of the study. That is to say that, at this stage, we are more interested in the utilization level of a cluster rather than its total size. Each serve can host up to **10** VMs. This number is also kept constant throughout the study. It also implies that the size of VMs is assumed to be constant.

There are a number of services running in this cluster. These services employ different VM level dependability techniques. Thus when a service is started in the cluster, it is tagged by the technique it uses. Hence the standbys and priorities of the services are set accordingly. Therefore, it can be rightly stated that the simulator simulates different VM level techniques and can be compared with the Markov diagrams developed in Section 5.2. The use of simulator relaxes a number of assumption made while developing the Markov models. The foremost advantage of using the simulator is that by doing so, the scope of study has raised from a single VM (in case of Markov models from Section 5.2) to a large number VMs hosting same type of service.

The simulator also consists failure processes for each type of failure. These failures include the physical and VMM failures of the server, application and OS failures of the VM and even operational failures. All of these failures have been aggregated in one single parameter while performing analytical modeling of the VM level dependability techniques, which is justifiable as all of them are n.e.d. processes [55]. But dealing with them individually gives more flexibility in the sense that their distributions can be changed or other customizations can be done with individual process. However, n.e.d. is the most suitable distribution to model time between failures [55] and hence these processes are considered to follow the same. The value of parameters for each of them in listed in Table 5.

6.4 Simulator Model

In order to develop the simulator which is used to perform the intended study, an activity diagram for it is sketched. This diagram is presented in Figure 36. As already stated that Simula/DEMOS are chosen for developing the simulator, the conceptual blocks of Simula/DEMOS are used in the activity diagram also. Hence some of the components are modeled as entities while others are modeled as resources (bins to be specific). It should be noticed that management system is not a part of the simulator. This leaves the management tasks to be performed by other entities.

In the following, a brief description of each component is highlighted. This will help in in-depth understanding of the activity diagram for simulator. It should be made clear that the activity diagram doesn’t show all minute details of each entity. Instead, it focuses only on salient features of an entity in relation to the intended study.

6.4.1 Service Deployers

This entity is there just to deploy a new service in the cluster. Hence, it performs the deployment task of the management system. For the current study, the deployment of services is limited to a certain upper limit of number of services of each type. It is quite static in the sense that there are no more services added after the upper limit is achieved, which is done quite rapidly. In the diagram it is marked as **Deployer**.

6.4.2 Service

This entity represents a service hosted by the cluster. It manages all the VMs employed by it. Thus the management tasks needed for deploying VM, performing failover, implementing dependability technique specific logic and collecting data is performed in this entity. As the different dependability techniques have quite different implementation logic, services employing each of them is modeled as a separate entity. Thus there are four different service entities. It also means that there are four different service deployers for each type of service. During simulating, it should be kept in mind that there are as many instances of this entity as the number of services, which makes the simulation runtime quite long. The service of each type is marked as **Service** followed by its type in the activity diagram of Figure 36.

6.4.3 Virtual Machine

In the simulator, a VM is also implemented as an entity. The main task which VM performs is that it hosts a service and allocates and reserves resources for itself. It is also prone to failures so each VM also initiates its failure processes. In the same course, it also responds to failures which are sent as interrupts to it. The response is nothing but giving the control back to the service entity which it hosted and allowing it to perform failover or any other management task it is supposed to perform depending upon the service type. There is only one VM entity as all the required tasks are easily managed in it. But, there are as many instances to this entity as the total number of VMs including all the replicas, dedicated and shared. The total number of VMs is actually what determines the utilization of the cluster. As there is a large number of VMs in the cluster, it makes the simulation run very slow. This entity is marked as **VirtualM** in the diagram.

6.4.4 Server

This entity simulates a physical server. Hence it furnishes resources to VMs. It also initiates failure processes for itself. When interrupted by either of them, it interrupts all VMs utilizing its resources, which, in return, interrupt the respective services as already mentioned. The repair (or more precisely replacement) of a server is also catered by this entity. Hence it accounts for a server replacement and as the new server takes the place of the failed one, the entity is not terminated, rather it is restarted from the beginning. As already stated earlier, the total number of servers is assumed to be 1000 in the current study. This entity is denoted as Server in the activity diagram of Figure 36.

6.4.5 Failure Processes

There are two types of failure processes in the simulator. One type is local to VMs while the other is for servers. Hence they have different scopes as indicated in Table 5. The failure processes are n.e.d. with parameter values as given in Table 5. Individual VMs and servers have their own failure processes. Also there are different types of failures, for instance CPU failure, HDD failure etc, and each failure type is modeled as separate entity.

6.4.6 Virtual Resources

These are the resources which a server possesses and can be utilized by a VM. As already stated, a single server can host up to 10 VMs. This actually means that a server has 10 virtual resources. It is modeled as a bin local to each server. Hence when a server is started it has 10 virtual resources and VMs acquire the resources from there. In case a server fails, the VMs utilizing its resources are also failed. This is modeled by sending an interrupt to VM and also VM giving back the resources it has acquired to the bin. Then server holds all the resources until it is replaced and back in the service. The bin is referred to as **VM** in the diagram.

6.4.7 Data Collection

In the simulation study, we are interested in availability for services of each type. In order to find availability, we need to have the values of downtime of each service. This data is collected by using the built in data collection tool called **tally**. There are four tallies used in the simulator to collect data for each service type separately.

6.5 Simulator Validation

The simulator hence developed is validated by two ways. The first method is to trace the simulation output for each event. In this way it is ensured that the simulator carries out correct steps at right instances of time and/or in response to the events they are meant to. By this method, the simulator is found to be working as required.

The second method used for validating the simulator is to run the simulator for very low cluster utilization value (26% in this case). The resulting availability values for different techniques are compared to the values obtained by solving Markov models with assumption that there are always enough resources for restarting a failed VM, presented in Table 6. This comparison is shown in Table 2.

Table 2: Comparison of Analytical and Simulated Availability Values for Validating the Simulator

Technique	Analytical Value	Simulation Value
Updated Dedicated Hot Standby	0.999971	0.999958
Updated Shared Hot Standby	0.999852	0.999702
Shared Cold Standby (High Priority Service)	0.996214	0,998703
Shared Cold Standby (Low Priority Service)	0.9021	0.998725

It is seen from the above table that the availability values for hot standbys doesn’t differ much in analytical approach and simulation study. Even for high priority cold standby, the difference is only 0.25%. But there is a large difference seen in the availability values of analytical model and simulation study for cold standby (low priority). This is because in the analytical study, the cluster utilization factor is considered while in the simulation, as the utilization is only 26%, the probability that this service will get preempted is practically equals to zero. This causes this service type to have as high availability as its high priority counterpart as both of them employ the same standby technique, the only difference is in their priorities.

The above performed analysis validates the simulator and gives us confidence in the results generated by it.

In the following the activity diagram for developing the simulator is given. It should be noticed that the diagram doesn't show all the four service entities. Instead it just gives the model for one service entity. The reason for not including the other service entities is that in their presence the diagram will be too complex and this will cost its readability. Also, it is not very important to have those service entities as the main flow of the simulator and entity interactions are clear even from the simplified version.

The source code file along with the modified DEMOS file is submitted along with this thesis for further reference.

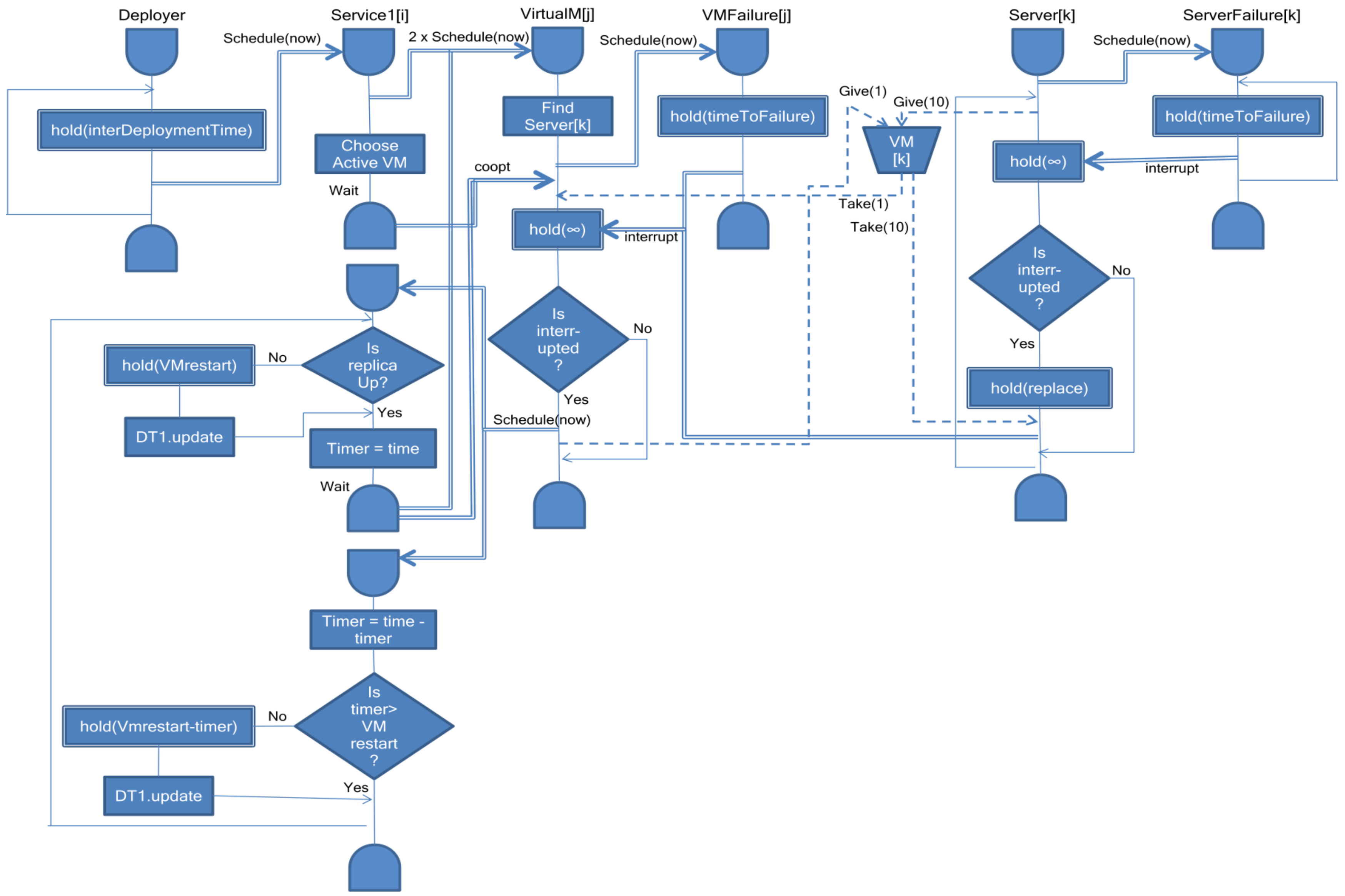

Figure 36: Activity Diagram for Simulator Development

7 Results and Discussions

Throughout the report, different objectives of the present work, which are outlined in Section 1.3, have been achieved at different places. For the sake of continuity, most of the results are already presented in appropriate places and their discussions are also performed. Hence the classification of services, both according to the user requirement and deployment nature is performed and discussed in Chapter 2. Similarly, classification of techniques for differentiating dependability of services is done and discussed thoroughly in Chapter 3. On the same course, the preliminary analysis and discussion of both the VM level dependability differentiation and cloud level differentiation is performed in Chapter 5.

In this chapter, some further results which are obtained from the models and simulations are presented and discussed. But before that, a summary of all the parameters used for numerical analyses is given.

Therefore, the results presented in this section include;

1. Comparison of service availability provided by different VM level techniques,
2. Comparison of service availability provided by combination of different VM level techniques and cloud level scenarios,
3. Checking sensitivity of service availability in relation to the availability of management system for all cloud level scenarios and both hot and cold standbys,
4. Analyzing service availability compared to the cluster load (using simulation)

7.1 Parameters

In this section, all the parameters used throughout the study are tabulated along with their values. These parameter values are found by extensive research in the literature and other web sources. It has been a tedious task as most of the cloud operators and other service providers do not make their failure data public. Thus not all of the parameters required to conduct the present study were obtained from documented sources. Some of the parameter values are thus approximated.

Two types of approximation methods for determining the parameter values which are not found reported are used in the current study. First, if a relationship is found between the parameter whose value is to be determined and a parameter whose value is known, then the parameter value is computed using this relationship. This method is used for finding value of operational failure intensity, for instance, by considering its relationship with software failures as indicated in [34]. In the following tables, this will be indicated as **calculated** in the **source** column. Second, in case not even a relationship with a known parameter is found, then a careful guess of the value is made. It is made sure that the value thus obtained is realistic and will produce reasonable results. This is indicated as a**ssumed** in the **source** column of parameter value tables.

In the following three different tables for parameter values are given, each for different study approach. Hence Table 3 gives values used for evaluating VM level models, Table 4 for Cloud level models and finally Table 5 for the simulator.

Table 3: Parameter Values for VM Level Models of Section 5.2

Name	Parameter	Value	Unit	Description	Source
VM Failure Rate	**λ**	0.00722	hr^{-1}	Aggregates all hardware, software and operational failures	[13] [65] [58] [34], Calculated
VM Restart Rate	**μ**	2.0	hr^{-1}		[13]
Standby Update Rate	**α**	60.0	hr^{-1}	Rate of updating a shared hot standby in case active fails	[36], Calculated
Cluster Boundary Recalculating Rate	**γ**	6.0	hr^{-1}	Rate of increasing new resources to the cluster in case the required number of resources is not achieved	Assumed
Preemption Rate	**δ**	0.2	hr^{-1}	Rate of preempting VM running cold standby (low priority) service	Assumed
VM Restart Coverage Factor	**c**	0.95		Coverage factor for restart of a new VM replica	[13]
Probability of Available Resources	**p_x**			Probability that enough resources are available to restart a new VM of type **x**	

Table 4: Parameters for Cloud Level Models of Section 5.3

Block Name	Parameter	Value	Definition	Source
Site Attributes	**A_{site}**	0.9975	Aggregate Availability of Site Attributes (Cooling + Power)	[64] [34]
PDU	**A_{pdu}**	0.9992	Availability of PDU	[75]
Management System	**A_{mngt}**	0.999	Availability of Management System (same values for service, cloud, datacenter and cluster layers)	Assumed
Server Matrix	**A_{server}**	Different values	Availability of Server (from VM Level Models)	Section 5.2
Standby Storage	**$A_{storage}$**	1.0	Availability of cold replica of VM stored in cloud storage	Section 5.3.2
Switch	**A_{swtich}**	0.97986	Avalailabilty of Switch	[76]
Router	**A_{router}**	0.99966	Avalailabilty of Router	[75]
Access Network	**A_{access}**	0.989	Availability of Access Network to the Datacenter	[77]
Core Network	**A_{core}**	0.999	Availability of Core Network (e.g. NIX for Norway)	[34]
User-side Access Network	**A_{user}**	0.99	Availability of Access Network on the User's Side	Assumed

In order to present the parameters used for simulation, it should be made clear that in the simulator, all failure and repair processes are simulated as n.e.d. It is a known fact that an n.e.d. depends upon only one parameter for its description [74]. Hence in Table 5, the values of this parameter for each process are given.

Table 5: Parameters for Simulator

Name	Parameter Value	Unit	Scope	Source
CPU Failure	1/2500000	hr^{-1}	Server	[13]
HDD Failure	1/142000	hr^{-1}	Server	[13]
Memory Failure	1/480000	hr^{-1}	Server	[13]
NIC Failure	1/120000	hr^{-1}	Server	[13]
Fan Failure	1/3100000	hr^{-1}	Server	[13]
Power Failure	1/670000	hr^{-1}	Server	[13]
VMM Failure	1/2880	hr^{-1}	Server	[13]
OS Failure	1/1440	hr^{-1}	VM	[13]
Application Failure	1/336	hr^{-1}	VM	[13]
Operational Failure	1/290	hr^{-1}	VM	[34], Calculated
VM Restart	2	hr^{-1}	VM	[13]
Server Replace	1	hr^{-1}	Server	[13], Calculated
Service Deployer	2	hr^{-1}	Service	Assumed

7.2 Comparison of VM Level Differentiation Techniques

In Section 5.2, the models for VM level differentiation in provision of availability to cloud services are drawn, solved, analyzed and commented upon. The effect of simplification of the models is also studied. The results obtained by solving these models for numerical values from Table 3 are summarized in Table 6.

Table 6: Availability Values for Different VM Level Dependability Techniques

Technique	Availability Value
Updated Dedicated Hot Standby	0.999971
Updated Shared Hot Standby	0.999852
Shared Cold Standby (High Priority Service)	0.996214
Shared Cold Standby (Low Priority Service)	0.9021

It is seen from the above table that the more updated the standby is, the higher is the availability of the service. Another observation is that the higher the availability of resources at the instant of failure, the larger is the service availability. This implies that updated standby with dedicated resources has the highest availability, but it is interesting to notice that it is immediately followed by updated shared hot standby. Although both of the techniques use hot standbys, which increase the possibility of failure in the service provision, they actually provide a very high availability value.

The fact that the updated shared hot standby guarantees 99.98% availability is really useful. Firstly, because it is not very far from the availability provided by updated dedicated hot standby and provides this high availability on comparatively very low cost (both in terms of resources and operation and maintenance cost). Secondly because, it is even better than the highest commercially provided guarantee of 99.95%, that is provided by Amazon Web Services [4].

On the other hand, it can be seen that there is a big lagging in the cold standby techniques compared to the hot ones. This is quite understandable because the time required for actually resuming the service is quite high in these techniques. It is again quite obvious that the low priority service has a very low availability. It is interesting though to note that even with a cold standby, an availability guarantee of as high has 99.5% is possible to be provided.

7.3 Comparison of Cloud Level Differentiation Techniques

In Section 5.3, block diagrams are drawn for presenting and analyzing different scenarios to provide dependability differentiation by spatial separation of active and standby replicas. Initial analyses of these models have resulted into simpler diagrams and less complex availability expressions for each scenario. In this section, a comparison of these scenarios is done. But, as already mentioned in Section 5.3, these diagrams depend greatly upon the VM level differentiation technique used as it constitutes a main parameter in availability expression.

Hence, the following comparison takes into consideration both the cloud level scenario and the underlying VM level technique for dependability differentiation. In order to come up with the availability values for different combinations, the parameter values for block diagrams are taken from Table 4 and the availability values of different VM level techniques of Table 6 are used. The results are shown in Table 7.

Table 7: Comparison of Different Combinations of Cloud Level Scenarios and VM Level Techniques

Cloud Level Scenario	VM Level Technique	Availability Value
Scenario I: All Replicas in the same Cluster of a Public Cloud	Updated Dedicated Hot Standby	0.981646
	Updated Shared Hot Standby	0.981529
	Shared Cold Standby (High Priority)	0.977957
	Shared Cold Standby (Low Priority)	0.885568
Scenario II: All Replicas in the same Datacenter of a Public Cloud	Updated Dedicated Hot Standby	0.983054
	Updated Shared Hot Standby	0.983052
	Shared Cold Standby (High Priority)	0.983048
	Shared Cold Standby (Low Priority)	0.982918
Scenario III: All Replicas in the same Public Cloud	Updated Dedicated Hot Standby	0.987004
	Updated Shared Hot Standby	0.987003
	Shared Cold Standby (High Priority)	0.986984
	Shared Cold Standby (Low Priority)	0.986482
Scenario IV: Replicas in Hybrid Cloud	Updated Dedicated Hot Standby	0.98798
	Updated Shared Hot Standby	0.987979
	Shared Cold Standby (High Priority)	0.987956
	Shared Cold Standby (Low Priority)	0.987363

It is observed from the above table that the best availability is provided if the service employs updated dedicated hot standby technique and is deployed in a hybrid cloud. This observation is very obvious. But it is also obvious that the cost of implementation of this method is very high. The cost is contributed by; hot standby, dedicated resources, updating mechanism and more than one cloud providers.

But, it is found that except for scenario I even updated shared hot standby provides as high availability as the dedicated hot standby does for respective scenarios. This suggests that shared hot standby is as good as dedicated hot standby is, not to mention the cost benefit which it gives. Hence, it can replace the latter for providing high availability to crucial services.

One interesting finding from Table 7 is that even shared cold standby (high priority) does not lag far behind the hot standby techniques, except for scenario I, and is best for scenario II. This suggests that if a cloud provider has only one datacenter (e.g. a company owning a private cloud for its employees), then almost same level of availability can be provided by simply having enough resources in the datacenter, so that it isn't overloaded, and keep the standbys unpowered in storage although in a different cluster.

7.4 Dependence of Service Availability on Management System

This is argued a number of times in this work that management system has a very crucial role to play in availability of services hosted in a cloud environment. It has been noticed that management system can actually be divided into four distinct layers managing VMs, resources and services on different levels of abstraction. Previous results are calculated assuming a constant value of availability of management system which is also given in Table 4. It is argued while using this value that it represents the availability of each layer of management system.

But, as management system is a very crucial part of the cloud service provision, it is worth checking the sensitivity of service availability against its availability. For this reason, all cloud level dependability scenarios for both hot standby and cold standby are checked against it. The results are shown in Figure 37 and Figure 38 respectively.

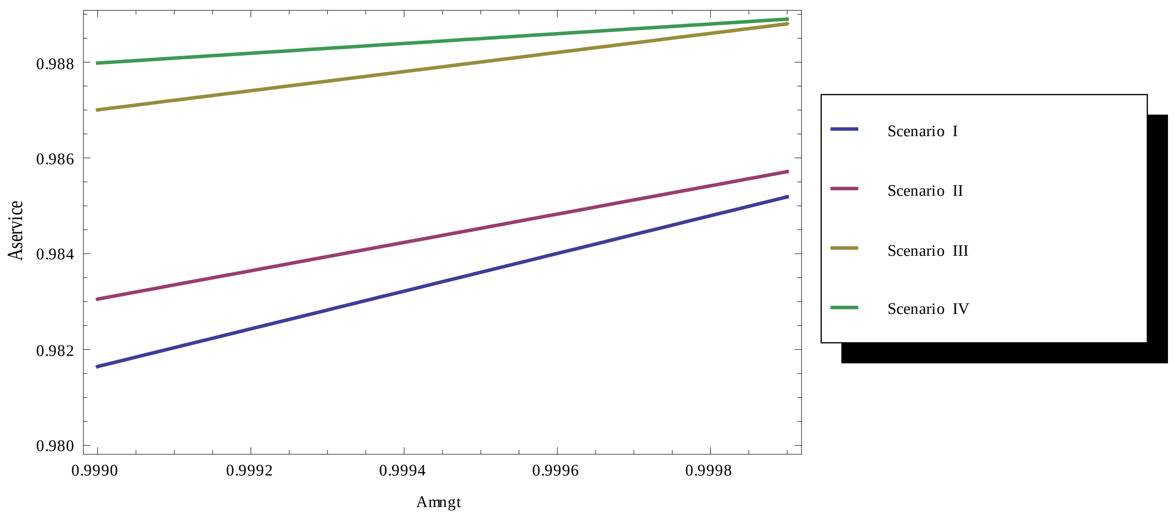

Figure 37: Service Availability VS Availability of Management System for Hot Standbys

It is observed from Figure 37 that a linear relationship exists between the service availability and availability of management system. As it is already observed in section 7.3 that scenario IV (replicas in hybrid cloud) provides the highest availability, the observation is visualized here. But it is rather interesting to note that for management system availability of **99.985%** scenario III (all replicas in the same cloud) also provides almost the same level of service availability. Hence it is suggested that if management system availability of around **four nines (99.99%)** is guaranteed, then some cost benefits can be achieved. Because then it will be possible to achieve the highest possible service availability by deploying the replicas in a single cloud (although different datacenters) instead of deploying them in a hybrid cloud.

Another observation made from Figure 37 is that the plot of scenario I (all replicas in the same cluster) has the highest slope among all. Hence for this scenario the service availability is rapidly increasing with increasing availability of management system and reaches very close to scenario II (all replicas in the same datacenter) at the management system availability value of **99.985%**.

The plot shown in Figure 38 does the same sensitivity check as it is done in Figure 37 but for the case when cold standby is employed.

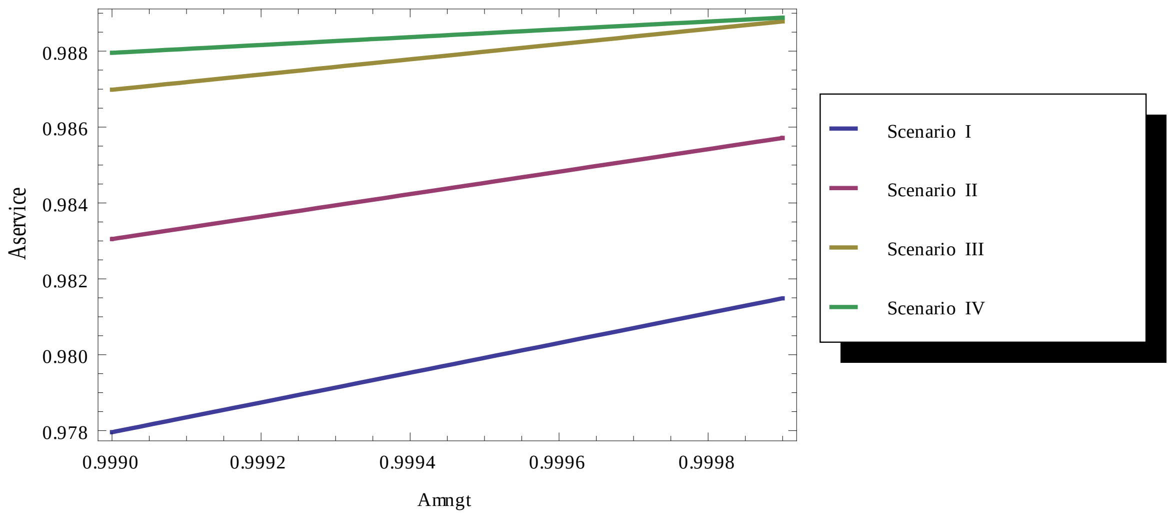

Figure 38: Service Availability VS Availability of Management System for Cold Standbys

It is seen from the above plot that the behavior of service availability in relation to the availability of management system is the same regardless the type of standby used. Hence the plots for different cloud level scenarios with cold standbys follow the same linear trend as it is the case for hot standbys as seen from Figure 37.

However, a difference lies in the value of service availability as hot standbys provide higher availability then cold standbys. Another difference lies in the slope of plot for scenario I (All replicas in the same cluster). It is seen that the plot for scenario I has higher slope than the others for hot standby. But for cold standby technique, the slope doesn't differ much from scenario II (All replicas in the same datacenter).

The general observation of having the lowest availability value for scenario I (all replicas in the same cluster) and the highest for scenario IV (replicas in a hybrid cloud) is caused by the fact that in former scenario, there are a lot of single points of failures while in the later there are the least. But interesting deviations from the norm have been seen in Figure 37, Figure 38, and Table 7 as already has been pointed out and discussed.

7.5 Dependence of Service Availability on Cluster Load

Although a rough idea of dependence of service availability upon cluster load is obtained from the graphs presented in Section 5.2, the analytical model could not be used to quantify this dependence. Hence a study by using simulator is conducted. The results are shown in Figure 39.

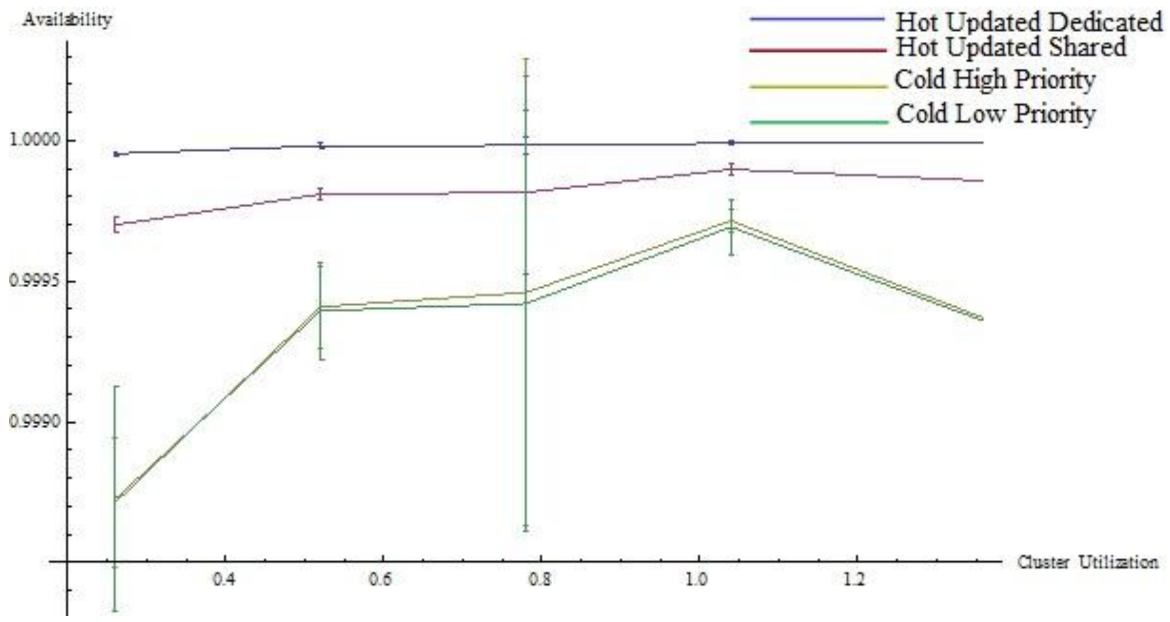

Figure 39: Service Availability VS Cluster Utilization

There are two salient observations made from the above figure for each plot, namely; the trend of service availability and the magnitude of 95% confidence interval.

For the case of hot updated dedicated standby, its availability remains almost constant regardless the load on cluster. This is justifiable because this type of service is given the highest priority. Even its confidence interval is very intact, this means that the availability is constant and its value remains almost stable throughout the trend line. It is very closely followed by shared updated hot standby. Although neither this technique has as good availability as the previous one, nor it is as stable as the previous one, but still, it gives quite high availability even for high cluster utilization rates.

On the other hand, cold standbys don’t only have low availability especially at the utilization rate of **136% (i.e. 36% overload),** but also show quite instability in availability values. This can be clearly seen by noticing the magnitudes of confidence intervals at different points. But, in case only mean values are considered, even these techniques have an ability to provide up to **99.9%** availability. But as already stated, confidence can’t be placed on these techniques due to huge margins of their confidence intervals.

One additional observation in this plot is that the value of cold standby (low priority) service is fairly higher than the value which is obtained from analytical models. The reason of this deviation is that although we previously deduced that the availability of this technique largely depends upon the cluster load, we had no means to find this dependence. Hence we just assumed a logical value. In this particular case, the simulation results have been used to validate analytical results as more confidence is put into simulation after verifying its traces and performing other validation checks.

8 Conclusion

8.1 Conclusion

In this thesis work, a number of achievements have been made. First of all, a classification of cloud services has been performed using two different criteria, which are; user's dependability requirement and nature of the service. These classifications are helpful because the former can be used to decide which services should be guaranteed higher dependability and prioritized in its deployment. While the latter classification is useful because it tells how exactly high dependability can be achieved. That is, it gives a hint of crucial resources needed to be replicated for the service and also suggest whether the standby replica should be updated or not.

Secondly, a tree of dependability differentiation techniques has been drawn and related techniques are grouped together. Combining the previous work done in the Autumn Project, a total of two different levels of dependability differentiation techniques are identified. The lower level is referred to as VM level in this thesis while the higher level is called the cloud level. Techniques in the former level actually depend upon the state of the standby replica while those in higher level depend upon the physical location of standby replica. Dependability differentiation is provided in the former case by having standby replicas with different states, e.g. shard or dedicated, hot or cold etc. While in the latter case, differentiation is provided by reducing the number of single points of failures, hence replicas are hosted in different clusters or datacenters or even clouds.

The third achievement made in this work is the development of a simulator which can simulate a cloud cluster for evaluating the dependability techniques on VM level. This tool is very useful especially in this case because the analytical models have enormous limitations in dealing with dynamic systems and cloud system has proven to be highly dynamic. The simulator developed in this work was also used to validate some assumptions made, specifically the rate of preemption of low priority service, in analytical models because simulator could deal with the dynamicity of the problem while the analytical models couldn't.

After performing different comparisons it can be concluded that the best VM level dependability differentiation technique is the one with **updated shared hot standby.** The reason for this judgment is that although it doesn't have as high availability as **updated dedicated hot standby** has, but it doesn't lag far behind. The added advantage in its favour is it being very economic, in the all senses from investment to management, and from operation to monitoring and maintenance. Actually in some cases, it is as good as **updated dedicated hot standby** is. For instance, if management system guarantees 99.99% availability, then the availability values for both of the techniques coincide. Similar coinciding can be found in case the two techniques are implementing scenario II (different clusters same datacenter) of cloud level differentiation. Similar effect is seen in the simulation study whereby **updated shared hot standby** guaranteed an almost constant and stable availability of around 99.97% regardless the load on the cluster.

It can also be rightly concluded that although there are a number of techniques for guaranteeing cloud dependability, network dependability still remain quite low. And as it is seen from the availability expressions, network availability has quite a weightage in them. Although some techniques of network dependability differentiation are already proposed and thoroughly researched [57], this work has to be done in relation to cloud services.

8.2 Future Work

The study conducted in this thesis is limited to two out of five parameters which account for service availability. Hence a more enhanced study which gathers the remaining three parameters also, that are; network performance, cloud performance and cloud security, has to be conducted.

Also, although a simulator is developed in this work, its scope is limited to the study of VM level differentiation, it can be enhanced to study the whole cloud system as the analytical models pose limitations to the dynamicity of cloud computing paradigm.

Even for this level of abstraction, simulator can be made more sophisticated by introducing more complex prioritization techniques then simple preemption and so on.

Again, although simulation study introduced dynamicity to the otherwise static study, some more dynamicity can be added to the simulator by introducing functionalities like live migration in case of cluster over load etc.

For analytical study also, the sensitivity of service availability can be checked against each component of cloud system and an optimum path set can be deduced out of it. Thus each component in the cloud system will give its input towards the higher availability.

Conclusively, it can be stated that the field of dependability differentiation in cloud services is rather new and there are leagues and leagues to be covered in order to make it mature and fruitful.

&&& %%% THE END %%% &&&

9 References

1. *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.* **Buyya, Rajkumar, et al.** 6, 2009, Future Generation Computer Systems, Vol. 25, pp. 599–616.

2. **Moorsel, Aad van.** Metrics for the Internet Age: Quality of Experience and Quality of Business. *Fifth Performability Workshop.* September 16, 2001.

3. *Performance and Availability of Internet Data Centers.* **Menascé, Daniel A.** 3, 2004, IEEE Internet Computing, Vol. 8, pp. 94-96.

4. Amazon EC2 Service Level Agreement. *Amazon Web Services.* [Online] October 23, 2008. [Cited: December 5, 2010.] http://aws.amazon.com/ec2-sla/.

5. **Helvik, Bjarne E.** *Dependable Computing Systems and Communication Networks: Design and Evaluation.* Trondheim, Norway : Tapir Akademisk Forlag, 2009.

6. Google Apps Service Level Agreement. *Google Apps.* [Online] [Cited: 12 6, 2010.] http://www.google.com/apps/intl/en/terms/sla.html.

7. Windows Azure Service Level Agreements. *Windows Azure: Microsoft's Cloud Services Platform.* [Online] [Cited: 12 6, 2010.] http://www.microsoft.com/windowsazure/sla/.

8. Products & Services. *Amazon Web Services.* [Online] Amazon. [Cited: 12 08, 2010.] http://aws.amazon.com/products/.

9. Apps for Business. *Google.* [Online] [Cited: 12 08, 2010.] http://www.google.com/apps/intl/en/business/index.html.

10. Windows Azure AppFabric. *Microsoft.* [Online] [Cited: 12 08, 2010.] http://www.microsoft.com/en-us/appfabric/azure/default.aspx.

11. **Undheim, Astrid, et al.** *Technical and Business Perspectives of Cloud Computing.* 2010. Telenor Report R9/2010.

12. **Chilwan, Ameen.** *Dependability Differentiation in Cloud Services.* Telematics, NTNU. 2010. Project Report.

13. *Availability Modeling and Analysis of a Virtualized System.* **Kim, Dong Seong, Machida, Fumio and Trivedi, Kishor S.** 2009. Proceedings of the15th IEEE Pacific Rim International Symposium on Dependable Computing. pp. 365-371.

14. *The NIST Definition of Cloud Computing.* **Mell, Peter and Grance, Tim.** 2009, National Institute of Standards and Technology, Information Technology Laboratory.

15. *A Break in the Clouds: Towards a Cloud Definition.* **Vaquero, Luis M, et al.** 1, 2009, ACM SIGCOMM Computer Communication Review, Vol. 39, pp. 50-55.

16. *The NIST Definition of Cloud Computing.* **Mell, Peter and Grance, Timothy.** s.l. : NIST Special Publication 800-145, 2011, National Institute of Standards and Technology, Information Technology Laboratory.

17. **Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia.** *Above the Clouds : A Berkeley View of Cloud Computing.* s.l. : University of California at Berkeley, 2009.

18. Virtualization Overview. *VMWare White Paper.* 2006.

19. *Data center evolution: A tutorial on state of the art, issues, and challenges.* **Krishna, Kant.** s.l. : Elsevier, 2009, Computer Networks.

20. **Dropbox.** Dropbox for Android. [Online] Dropbox. [Cited: 05 31, 2011.] https://www.dropbox.com/android.

21. **Holma, Harri and Toskala, Antti.** *WCDMA for UMTS: HSPA Evolution and LTE.* s.l. : John Wiley & Sons Ltd., 2010.

22. *Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities.* **Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros.** 2009. International Conference on High Performance Computing & Simulation.

23. **Armbrust, Michael, et al.** *Above the Clouds: A Berkeley View of Cloud Computing.* Electrical Engineering and Computer Science, University of California. 2009. Technical Report.

24. Virtual Private Cloud (VPC). *Amazon Web Services (AWS).* [Online] [Cited: 12 09, 2010.] http://aws.amazon.com/vpc/.

25. *Toward a Unified Ontology of Cloud Computing.* **Youseff, Lamia, Butrico, Maria and Da Silva, Dilma.** 2008. Grid Computing Environments Workshop. pp. 1-10.

26. *What's Inside the Cloud? An Architectural Map of the Cloud Landscape.* **Lenk, Alexander, et al.** 2009. ICSE Workshop on Software Engineering Challenges of Cloud Computing.

27. *pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems.* **Verma, Akshat, Ahuja, Puneet and Neogi, Anindya.** 2008. Proceedings of the 9th International Conference on Middleware. Vol. 5346, pp. 243-264.

28. *Ant System for Service Deployment in Private and Public Clouds.* **Csorba, Máté J., Meling, Hein and Heegaard, Poul E.** 2010. Proceeding of the 2nd workshop on Bio-inspired algorithms for distributed systems.

29. *Live Migration of Virtual Machines.* **Clark, Christopher, et al.** 2005. Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. Vol. 2, pp. 273-286.

30. *Live Migration of Virtual Machine Based on Full-System Trace and Replay.* **Hai Jin, Haikun Liu, Xiaofei Liao, Liting Hu, Peng Li.** 2009. Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing.

31. *Black-box and Gray-box Strategies for Virtual Machine Migration.* **Wood, Timothy, et al.** 2007. 4th USENIX Symposium on Networked Systems Design & Implementation. pp. 229-242.

32. *Cost-Efficient and Differentiated Data Availability Guarantees in Data Clouds.* **Bonvin, Nicolas, Papaioannou, Thanasis G and Aberer, Karl.** 2010, In Proceedings of the ICDE, pp. 1-4.

33. *Dynamic Cost-Efficient Replication in Data Clouds.* **Bonvin, Nicolas, Papaioannou, Thanasis G. and Aberer, Karl.** 2009. Proceedings of the 1st workshop on Automated control for datacenters and clouds. pp. 49-56.

34. **Barroso, Luiz A and Hölzle, Urs.** *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Google Inc. 2009. Synthesis Lectures on Computer Architecture.

35. **VMware.** *Protecting Mission-Critical Workloads with VMware Fault Tolerance.* 2009. White Paper.

36. *Remus: High Availability via Asynchronous VirtualMachine Replication.* **Brendan Cully, Geoffrey Lefebvre, Dutc hMeyer,Mike Feeley, Norm Hutchinson, and Andrew Warfiel.** 2008. Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation.

37. **Louth, William.** Metering != Billing, Metering > Billing. *The Art of API Design and Performance Engineering.* [Online] 03 15, 2010. [Cited: 06 04, 2011.] http://williamlouth.wordpress.com/2010/03/15/metering-billing-metering-billing/.

38. **JINSPIRED.** Metering the Cloud. *JINSPIRED - Metering the Cloud.* [Online] [Cited: 06 05, 2011.] http://www.jinspired.com/products/jxinsight/meteringthecloud.html.

39. **Patel, Japan.** Billing and Metering in a Cloud. *Evolution and Awakening.* [Online] 05 24, 2011. [Cited: 06 06, 2011.] http://radiantlabs.wordpress.com/2011/05/24/billing-and-metering-in-a-cloud/.

40. *Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds.* **Elmroth, Erik and Larsson, Lars.** 2009. Eighth International Conference on Grid and Cooperative Computing. pp. 253-260.

41. **OpenNebula.** Featured Users. *OpenNebula: The Open Source Toolkit for Cloud Computing.* [Online] [Cited: 06 24, 2011.] http://www.opennebula.org/community:users.

42. *Virtual Infrastructure Management in Private and Hybrid Clouds.* **Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente, Ian Foster.** 5, 2009, IEEE Internet Computing, Vol. 13, pp. 14-22.

43. **Google.** Google App Engine. [Online] [Cited: 06 25, 2011.] http://code.google.com/appengine/.

44. **Microsoft.** Microsoft Windows Azure. [Online] [Cited: 03 10, 2011.] http://www.microsoft.com/windowsazure/windowsazure/default.aspx.

45. **Mazhar, Jamal.** Application Centric Vs. Infrastructure Centric Management of Resources. *Kaavo.* [Online] 11 25, 2008. [Cited: 06 25, 2011.] http://www.kaavo.com/blog/-/blogs/application-centric-vs--infrastructure-centric-management-of-resources.

46. —. More on Application Centric vs. Infrastructure Centric Management. *Kaavo.* [Online] 01 16, 2009. [Cited: 06 25, 2011.] http://www.kaavo.com/blog/-/blogs/more-on-application-centric-vs--infrastructure-centric-management?_33_redirect=%2Fblog.

47. **Kaavo.** High-Level Overview of an End-to-End Cloud Solution Architecture. *Kaavo.* [Online] 12 03, 2010. [Cited: 06 25, 2011.] http://www.kaavo.com/cloud-architecture.

48. **Google.** Google Apps for Business. [Online] [Cited: 03 10, 2011.] http://www.google.com/apps/intl/en/business/index.html.

49. **Amazon.** Amazon Web Services. [Online] [Cited: 03 10, 2011.] http://aws.amazon.com/.

50. **Salesforce.** Sales Cloud. *Salesforce CRM.* [Online] [Cited: 03 10, 2011.] http://www.salesforce.com/eu/crm/sales-force-automation/?d=70130000000FWoa&internal=true.

51. **Microsoft.** Microsoft SQL Azure. [Online] [Cited: 03 10, 2011.] http://www.microsoft.com/en-us/sqlazure/default.aspx.

52. **Salesforce.** Service Cloud. *Salesforce.* [Online] [Cited: 06 25, 2011.] http://www.salesforce.com/eu/crm/customer-service-support/.

53. *Dynamo: Amazon's Highly Available Key-value Store.* **Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels.** 2007. Symposium on Operating Systems Principles - SOSP '07. Vol. 21.

54. *Basic Concepts and Taxonomy of Dependable and Secure Computing.* **Avizienis, Algirdas, et al.** 1, 2004, IEEE Transactions on Dependable and Secure Computing, Vol. 1, pp. 11-33.

55. **Emstad, Peder J, et al.** *Dependability and Performance in Information and Communication Systems.* Trondheim, Norway : Tapir Akademisk Forlag, 2009.

56. **Semeria, Chuck and Stewart, John W. III.** *Supporting Differentiated Service Classes in Large IP Networks.* s.l. : Juniper Networks, 2001. White Paper.

57. *A Survey of Resilience Differentiation Frameworks in Communication Networks.* **Cholda, Piotr, et al.** 4, 2007, IEEE Communications Surveys & Tutorials, Vol. 9.

58. *Failure Trends in a Large Disk Drive Population.* **Pinheiro, Eduardo, Weber, Wolf-Dietrich and Barroso, Luiz A.** 2007. 5th USENIX Conference on File and Storage Technologies.

59. *A Self-Organized, Fault-Tolerant and Scalable Replication Scheme for Cloud Storage.* **Bonvin, Nicolas, Papaioannou, Thanasis G. and Aberer, Karl.** 2010. International Conference on Management of Data. pp. 205-216.

60. **Peters, Mark.** *Implementing the Right High Availability and Disaster Recovery Plan for Your Business.* Enterprise Strategy Group, Symantec. 2010. White Paper.

61. **Symantec.** *Veritas Cluster Server Implementation Guide - ESX.* 2011. Guide. http://sfdoccentral.symantec.com/sf/5.1MP2/esx/pdf/vcs_implementation.pdf.

62. **VMWare.** *VMware High Availability: Concepts, Implementation and Best Practices.* 2007.

63. *Power Provisioning for a Warehouse-sized Computer.* **Fan, Xiaobo, Weber, Wolf-Dietrich and Barroso, Luiz André.** 2007. roceedings of the 34th annual international symposium on Computer architecture. pp. 13-23.

64. **W. Pitt Turner IV, J. H. Seader, and K. G. Brill.** Tier Classifications Define Site Infrastructure Performance. *The Uptime Institute White Paper.* 2006.

65. *Characterizing Cloud Computing Hardware Reliability.* **Vishwanath, Kashi V and Nagappan, Nachiappan.** Indianapolis, Indiana, USA : s.n., 2010. Proceedings of the 1st ACM symposium on Cloud computing.

66. *Fighting Bugs: Remove, Retry, Replicate and Rejuvenate.* **Grottke, Michael and Trivedi, Kishor S.** 9, s.l. : IEEE Computer Society, 2007, Vol. 43.

67. *Why do internet services fail, and what can be done about it?* **Oppenheimer, David, Ganapathi, Archana and Patterson, David A.** 2003. USITS'03 Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems. Vol. 4.

68. **Telematics, Department of.** Mathematica. *Courses and themes - TTM4120.* [Online] [Cited: 06 30, 2011.] http://www.item.ntnu.no/fag/ttm4120/current/mathematica.php.

69. **Chris Black, Ajay Chandramouly, Vishwa Hassan, Ashish Khare, Bharat Mohla, Sanjay Rungta, Terry Yoshi.** Virtualizing Mission-Critical Applications. *IT@Intel White Paper*. January 2011.

70. **Baruah, Ayushman.** Virtualization helps PVM improve hardware utilization by 70 percent. *Information Week.* [Online] 04 25, 2011. [Cited: 06 30, 2011.] http://www.informationweek.in/Data_Center/11-04-25/Virtualization_helps_PVM_improve_hardware_utilization_by_70_percent.aspx.

71. **Zukerman, Moshe.** *Introduction to Queueing Theory and Stochastic Teletraffic Models.* 2010. Lecture Notes.

72. *Application of the RESTART/Splitting Technique to Network Resilience Studies in NS2.* **Anders Mykkeltveit, Bjarne E. Helvik.** 2008. Proceeding (620) Modelling and Simulation.

73. Simula/Cim. *Norwegian University of Science and Technology.* [Online] [Cited: 12 19, 2010.] http://www.item.ntnu.no/fag/ttm4110/current/simula.php.

74. **Birtwistle, Graham.** *DEMOS - A system for Discrete Event Modelling on Simula.* s.l. : University of Sheffield, 2003.

75. *Designs, Lessons and Advice from Building Large Distributed Systems.* **Dean, Jeff.** 2009. Keynote from LADIS.

76. *VL2: A Scalable and Flexible Data Center Network.* **Greenberg, Albert, et al.** 2009. Proceedings of the ACM SIGCOMM 2009 conference on Data communication.

77. *End-to-End WAN Service Availability.* **Dahlin, Michael, et al.** 3, 2003, IEEE/ACM Transactions on Networking (TON), Vol. 11.