UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

A Fault-Tolerant Mechanism for Desktop Cloud Systems

by

Abdulelah Alwabel

Thesis for the degree of Doctor of Philosophy in Computer Science

June 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AN ENGINEERING

Electronics and Computer Science

Doctor of Philosophy

A FAULT-TOLERANT MECHANISM FOR DESKTOP CLOUD SYSTEMS

Abdulelah Alwabel

Cloud computing is a paradigm that promises to move IT another step towards the age of computing utility. Traditionally, Clouds employ dedicated resources located in data centres to provide services to clients. The resources in such Cloud systems are known to be highly reliable with a low probability of failure. Desktop Cloud computing is a new type of Cloud computing that aims to provide Cloud services at little or no cost. This ambition can be achieved by combining Cloud computing and Volunteer computing into Desktop Clouds, harnessing non-dedicated resources when idle.

The resources can be any type of computing machine, for example a standard PC, but such computing resources are renowned for their volatility; failures can happen at any time without warning. In Cloud computing, tasks are submitted by Cloud users or brokers to be processed and executed by virtual machines (VMs), and virtual mechanisms are hosted by physical machines (PMs). In this context, throughput is defined as the proportion of the total number of tasks that are successfully processed, so the failure of a PM can have a negative impact on this measure of a Desktop Cloud system by causing the destruction of all hosted VMs, leading to the loss of submitted tasks currently being processed. The aim of this research is to design a VM allocation mechanism for Desktop Cloud systems that is tolerant to node failure. VM allocation mechanisms are responsible for allocating VMs to PMs and migrating them during runtime with the objective of optimisation, yet those available pay little attention to node failure events.

The contribution of this research is to propose a Fault-Tolerant VM allocation mechanism that handles failure events in PMs in Desktop Clouds to ensure that the throughput of Desktop Cloud system remains within acceptable levels by employing a replication technique. Since doing so causes an increase of power consumption in PMs, the mechanism is enhanced with a migration policy to minimise this effect, evaluated using three metrics: throughput of tasks; power consumption of PMs; and service availability. The evaluation is conducted using DesktopCloudSim, a tool developed for the purpose by this study as an extension to CloudSim, the well-known Cloud simulation tool, to simulate node failure events in Cloud systems, analysing node failure with real data sets of collected from Failure Trace Archives. The experiments demonstrate that the FT mechanism improves the throughput of Cloud systems statistically significantly compared with traditional mechanisms (First Come First Serve, Greedy and RoundRobin) in the presence of node failures. The FT mechanism reduces power consumption statistically significantly when its migration policy is employed.

Table of Contents

List of Tables

List of Figures

DECLARATION OF AUTHORSHIP

I, Abdulelah Alwabel, declare that this thesis entitled ‘A Fault-tolerant Mechanism for Desktop Cloud Systems’ and the work presented in it is my own and has been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- Parts of this work have been published (see section 1.3 for a detailed list).

Signed: ....

Date: ....

Acknowledgements

First and foremost. I should like to express my deep gratitude to my supervisors, Gary Wills and Robert (Bob) Walters, for having me as a PhD student under their supervision and their advice, discussions and, above all, unconditional support to complete this research. Their personal guidance has been essential through these past years.

I should also like to thank my friends in Southampton, with whom I have spent a great deal of time exchanging knowledge, discussing ideas and helping each other in tough times.

It is my honour to express my sincere appreciations to my parents, Abdulaziz and Sarah, for their unlimited support – not only to accomplish this thesis, but throughout my entire life. Abdulaziz and Sarah, without your help I really doubt if I could come close to completing this work.

My thanks go to my siblings, Abdullah, Norah, Abdulmalik and Abdulnasser, for their help and support.

Finally, I would like to thank Shahd, my future wife, for being there to help and encourage me.

Definitions and Abbreviations

BF	Best Fit
CSP	Cloud Service Provider
Eft	Estimated finish time
FCFS	First Come First Serve
FF	First Fit
FFD	First Fit Decreasing
FT	Fault-Tolerant
FTA	Failure Trace Archive
GB	Gigabyte
HPC	High-Performance Computing
IaaS	Infrastructure as a Service
K-S	Kolmogorov-Smirnov
kWh	Kilowatt-hours
LF	Least Fit First
MB	Megabyte
MI	Million Instructions
MIPS	Million Instructions Per Second
NF	Next Fit
OVF	Open Virtualisation Format
P2P	Peer-to-Peer
PaaS	Platform as a Service

PDF	Probability Distribution Function
PM	Physical Machine
QoS	Quality of Service
SaaS	Software as a Service
SE	Standard Error
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPEC	Standard Performance Evaluation Corporation
Std Dev	Standard Deviation
UBMP	Utilisation-Based Migration Policy
Var	Variance
VM	Virtual Machine
VMM	Virtual Machine Monitor

Chapter 1: Introduction

Providing computing power as a utility represents a research ambition that dates back to the 1960s [1], when John McCarthy predicted that computation would become a service managed and delivered so consumers had ready access and would pay only for its use. Several attempts, such as that by Grid and Clustering computing, have since been developed in the hope of achieving this ambition, yet none has so far succeeded.

The recent emergence of Cloud computing promises to achieve this vision through changing the IT sector for small and medium businesses so they are governed by giant third-party IT companies. Cloud computing is used by commercial companies to gain profit by selling their computing power to the public. It derives from Grid computing [2] and, traditionally, a Cloud service provider (CSP) such as Amazon EC2 [3] employs computing resources located in data centres to provide services to clients. The resources are dedicated to providing these services; that is, they are made for the purpose, with a valid claim to be highly reliable [4]. This type of Cloud computing is referred to as Traditional Clouds throughout this study.

However, the cost of operating Cloud services can be a barrier to many projects aiming to reap the benefits of Clouds, for example research projects where the budget is limited. Desktop Cloud computing has emerged to fill this gap with the aim of acquiring Cloud services from non-dedicated resources – if not for free, then at lower cost. It is a new type of Cloud computing that aims to provide Cloud capability by harnessing computing resources that would otherwise lie idle. This ambition can be realised by combining Cloud and Volunteer computing to form Desktop Cloud systems.

The main concept of Desktop Cloud computing is the use of computing resources to form an infrastructure containing a number of physical machines (PMs), which are then exploited to offer services based on the Cloud business model to end users. These PMs can comprise any computing resources, from normal PCs to servers. These can be used in a Desktop Cloud system when they are idle, determined by when their owners decide to join or leave the system. The concept is motivated by the success of Desktop Grid projects such as SETI@home [5], harnessing existing computing resources instead of tailor-made resources to form a Grid system and, according to Kondo et al., [6], this type of system is a Desktop Grid. The new concept combines the term 'Desktop' with that of 'Cloud', borrowed from Traditional Clouds. Throughout this study, a Desktop Cloud is a system that uses non-dedicated resources.

Desktop Clouds have some advantages over Traditional Clouds. The latter have a negative impact on the environment, since data centres consume massive amounts of electricity, a large part of which is for cooling [7]. Second, Desktop Clouds are more cost effective both for CSPs and consumers; there is no need to build further data centres to meet future demand, and services, if not free, are accessed more cheaply than those of Traditional Clouds. Moreover, they help to reduce energy consumption since they use already-running non-dedicated resources that would otherwise be idle. Studies have shown that, on average, the proportion of local resources lying idle within an organisation is about 80% [8]. Furthermore, Traditional Clouds are formed from a limited number of data centres around the globe and are therefore inefficient in terms of data mobility, paying little attention to clients' location [9]. Because they are centralised, potentially there could be a single point of failure if a provider goes out of business. By contrast, Desktop Clouds offer and manage decentralised services.

1.1 Research Motivation

In Cloud computing, virtual machine (VM) instances are hosted and run in PMs, so if a PM fails all allocated VM instances to it are destroyed. Consequently, a PM failure means that all the tasks running on the VM instance are lost, badly affecting throughput. Throughput refers to the number of tasks submitted by users that are successfully completed by a Desktop Cloud system. The problem of node failure can be addressed by developing a suitable VM allocation mechanism, defined as any technique or policy to manage the placement of VMs to PMs in Cloud systems [10], and involves the option of migrating a VM or group of VMs from the hosted PM to another to optimise Cloud systems by reducing energy requirements and power consumption, or enhancing performance throughput.

In Desktop Clouds, the problem of node volatility is crucial, because the number of node failure events may be high [11], so the goal of VM allocation in this context is to design a mechanism able to improve the throughput of a Desktop Cloud system in the presence of node failure. This research is motivated by the fact that it is considered to be the first attempt to tackle this issue by developing a novel mechanism for Desktop Cloud systems. The problem of VM allocation can be defined in Desktop Cloud as follows:

> *Given n number of PMs available to host, how to allocate m number of VMs into n PMs in a way that reduces the effect of failure events on efficiency of Desktop Clouds.*

This research develops a novel Fault-Tolerant (FT) VM allocation mechanism that can handle node failure by replicating VM instances, enhanced by developing a novel policy that migrates VM instances between PMs during runtime to reduce power consumption. The mechanism is evaluated by throughput, power consumption and availability metrics. Throughput metrics measure the number of successfully executed tasks submitted by users to a Desktop Cloud system; power consumption metrics measure the amount of energy consumed by PMs in a Desktop Cloud system; and availability metrics measure the computing power of a Desktop Cloud system that is available to serve users' new requests.

1.2 Research Hypotheses

The aim of this research is to improve the outcomes of Desktop Cloud systems, mainly in terms of throughput and power consumption. In order to achieve this goal, three research hypotheses are tested:

- Hypothesis H1: *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds.*
- Hypothesis H2: *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud system.*
- Hypothesis H3: *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with an accepted decrease in the throughput outcome.*

There are three main experiments that are conducted to test the aforementioned hypotheses, the experiments are:

- *Experiment I: Impact of Node Failure*. Conducted to demonstrate that throughput, power consumption and availability metrics can evaluate the impact of node failure on the outcome of Desktop Clouds. It investigates which of First Come First Serve (FCFS), Greedy, RoundRobin and Random VM allocation mechanisms yields the best results for each metric. The experiment tests the hypothesis H1.
- *Experiment II: Evaluation of the FT Mechanism.* Evaluates the FT VM allocation mechanism's improvement of the throughput of a Desktop Cloud system over other VM mechanisms tested in the first experiment. It compares the FT mechanisms in terms of the best results in each evaluation in the previous experiment and tests the hypothesis H2.

- *Experiment III: Utilisation-Based Migration Policy*. Investigates a way to reduce power consumption of nodes when the FT mechanism is employed. The mechanism implements a replication technique to ensure that, if a VM instance is destroyed when its hosted PM fails, there is a working copy on another PM. However, this technique can lead to nodes consuming more energy. The experiment tests the hypothesis H3.

Another experiment was conducted to show that the DesktopCloudSim tool developed for this study is capable of simulating node failure, comparing the results of three VM mechanism (FCFS, Greedy and RoundRobin) employed in either a Desktop Cloud or a Traditional Cloud. Such an experiment can illustrate the difference in outcomes between Desktop and Traditional Clouds in terms of throughput, power consumption and availability.

1.3 Contributions and Publications

This research focuses on the problem of the highly volatile nature of nodes in order to develop a mechanism that reduce the impact of node failure. The major contributions of this research are:

- *FT Mechanism*. This research developed a novel VM allocation mechanism to be used by Cloud management middleware in Desktop Cloud systems. The mechanism is tolerant of node failure because it ensures that the number of lost submitted tasks is less than under the traditional VM allocation mechanisms available in the literature and used in open source Cloud management middleware software such as Eucalyptus [12]. The mechanism employs a replication technique to replicate running VM instances and allocate them to various PMs, so if a VM is destroyed because of a failing PM, there is a replicated VM ready to take its place.

- *Utilisation-Based Migration Policy*. The FT mechanism is enhanced in terms of power consumption by a novel policy, referred to as the Utilisation-Based Migration Policy (UBMP), to migrate VM instances between PMs to reduce the energy consumed by PMs in a Desktop Cloud system. The idea is to migrate VM instances from PMs with low utilisation to PMs with higher utilisation. PMs with zero utilisation are put into power saving mode, because it has been demonstrated by [13] that an idle (i.e. with zero utilisation) machine may consume about 70% of the power used when fully utilised. Therefore, it is wise to develop the UBMP in order to minimise power consumption by Desktop Clouds' PMs, given that the FT mechanism can cause it to increase because of the replication technique.

In addition to the aforementioned major contributions, this research provides practical contributions as follows:

- *Analysis of Node Failure*. This research analyses the number of nodes expected to fail in a Desktop Cloud system during a given time, based on failure traces collected from online repositories, which record failure events in various Desktop Grid systems. This can indicate the expected number of failures in Desktop Cloud systems because, as this research shows, PMs in Desktop Grids and Desktop Clouds are similar.

- *Evolution Metrics*. This research proposes three metrics − throughput, power consumption and availability − to evaluate various VM allocation mechanisms. The throughput metric measures the number of successfully executed tasks submitted by users to a Desktop Cloud system; the power consumption metric measures the amount of energy consumed by PMs in a Desktop Cloud system; and the availability metric measures the amount of computing power of a Desktop Cloud system available to serve users' new requests. An experiment is conducted to demonstrate that these evaluation metrics are able to assess and compare the outcomes of various VM allocation mechanisms, namely FCFS, Greedy and RoundRobin. The main objective is to demonstrate the need to develop a novel VM allocation mechanism able to deal with the challenge of node failure.

- *DesktopCloudSim*. CloudSim [14] is a well-known and much-used simulation tool to simulate Cloud systems [15]. However, it lacks any ability to simulate node failure, which makes it unsuitable for Desktop Cloud systems. Therefore, this research extended CloudSim, developing DesktopCloudSim to simulate node failure in both Desktop and Traditional Cloud systems. It is stable and available online[1] for use by other researchers and, although developed mainly to simulate node failure in Desktop Clouds, it can also be used to simulate node failure in Traditional Clouds. In addition, it enables dynamic joining and leaving of PMs in a system during runtime, a feature missing from CloudSim.

[1] http://github.com/Abdulelah7/DesktopCloudSim

- *Traditional Clouds vs. Desktop Clouds*. This study conducted an empirical experiment using DesktopCloudSim to compare outcomes of Desktop and Traditional Cloud systems by means of the proposed evaluation metrics.

As mentioned, this research has published the following papers:

1. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary Brian (2015), Evaluation Metrics for VM Allocation Mechanisms in Desktop Clouds. In, *Emerging Software as a Service and Analytics (ESaaSA 2015)* accepted.

2. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary (2015), DesktopCloudSim: simulation of node failures in the cloud. In, The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization (*Cloud Computing* 2015), Nice, Fr, 22-27 Mar 2015.

   This was awarded a top paper prize in the *Cloud Computing* 2015 conference and was invited to be extended as a journal paper for the *International Journal on Advances in Software*.

3. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary B. (2015), A resource allocation model for desktop clouds. In, Chang, V., Walters, R. and Wills, G. (eds), *Delivery and Adoption of Cloud Computing Services in Contemporary Organizations*. Hershey, US, IGI Global, pp. 199-218.

4. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary (2014), Evaluation of Node Failures in Cloud Computing Using Empirical Data. In, *Open Journal of Cloud Computing* (OJCC), 1, (2), 15-24.

5. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary Brian (2014), A View at Desktop Clouds. In, *Emerging Software as a Service and Analytics (ESaaSA 2015) 2014*, Barcelona, ES, 03-05 Apr 2014, pp. 55-61.

6. Almutiry, Omar, Wills, Gary, Alwabel, Abdulelah, Crowder, Richard and Walters, Robert John (2013), Toward a framework for data quality in cloud-based health information system. In, *2013 International Conference on Information Society (i-society),* 24-26 Jun 2013, 153-157.

7. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary (2012), Towards a volunteer cloud architecture. In, *UK Performance Engineering Workshop* (*UKPEW 2012)*, Edinburgh, GB, 2pp.

8. Alwabel, Abdulelah, Walters, Robert John and Wills, Gary (2012), Towards performance evaluation in Volunteer Clouds. In, *European Conference on Service-Oriented and Cloud Computing (ESOCC2012)*, 19-21 Sep.

1.4 Thesis Outline

This thesis is divided into eight chapters. This first provides an overview of the research motivation, the hypotheses research tested and its contribution to the field. The remainder of this thesis is organised as follows:

- Chapter 2 presents an overview of Desktop Cloud computing and related computing areas. It starts by explaining the paradigm of Cloud computing. Grid computing is introduced next, being considered its forebear; understanding the difference between Cloud and Grid computing can help in understanding Desktop Clouds. Because the Desktop Grid is discussed as a special type of Grid system that is built on volunteer resources, it is beneficial to compare Desktop Cloud systems to Cloud, Grid and Desktop Cloud systems. The chapter concludes with research challenges in the area of Desktop Clouds. Node failure is introduced as the main research focus, addressed by implementing a suitable VM allocation mechanism.

- Chapter 3 reviews available VM allocation mechanisms developed for either Traditional or Desktop Cloud systems. The chapter explains that the VM allocation mechanism issue is a concern for research to improve the outcome of a Cloud system. The improvement can be either to reduce power consumption, improve performance or reduce the impact of data transfer of network. The chapter demonstrates that little attention has been paid to the problem of node failure in Cloud computing and concludes by stating that the solutions and mechanisms reviewed lack any ability to deal with the challenge to throughput of node failure in Desktop Clouds.

- Chapter 4 proposes a novel mechanism to replicate and allocate VMs to PMs in a way that enhances the resilience of Desktop Cloud systems against failure. The mechanism employs a replication technique to ensure that a replica is ready if a VM is destroyed because of node failure. It is enhanced by the development of a new policy that migrates VM instances during runtime in order to reduce power consumption. Several evaluation metrics are presented in the chapter to assess the outcomes of the proposed mechanisms to compare them.

- Chapter 5 presents the methodology used by this research to evaluate the proposed mechanism to show it is able to deal with the issue of node failure in Desktop Cloud systems. Simulation is used to conduct experiments, using DesktopCloudSim, a simulation tool developed by this research as an extension to CloudSim to model node failure in Desktop Cloud systems. The chapter presents an analysis of the number of failures in the PMs of two Desktop Grid systems to formulate an understanding of the expected level of failure in Desktop Cloud systems.

- Chapter 6 discusses the results obtained from the three experiments to test the hypotheses of this research. Each is tested and discussed separately. The first demonstrates how node failure affects throughput in Desktop Clouds when using the FCFS, Greedy, RoundRobin and Random VM mechanisms. The second demonstrates that the FT VM allocation mechanism improves throughput. The third demonstrates that a utilisation-based migration policy can reduce the power consumption of nodes in Desktop Cloud systems.

- Chapter 7 presents a discussion of the findings, then positions them alongside related findings in the literature, to illustrate that this study's contribution is to fill gaps in the research into Desktop Cloud computing. It starts by discussing the impact of node failure in Desktop Clouds, with an analysis of the number of nodes that can fail at any given time in both private and public Desktop Cloud systems. Next comes a discussion of the difference between Desktop and Traditional Cloud systems, using the DesktopCloudSim simulation tool. The proposed evaluation metrics are discussed in terms of their ability to assess the VM allocation mechanisms. The chapter next discusses the capacity of the designed FT mechanism to tolerate node failure in Desktop Cloud systems and to reduce power consumption. Lastly, it discusses the limitations of the research.

- Chapter 8 concludes the study with an outline of the research findings and contributions. It presents insights into the directions along which this research can be extended in future, and finishes with some final remarks about this research.

Chapter 2: Background

This chapter provides the study with essential background to Desktop Cloud computing, a new type of Cloud computing that employs the concept of the Desktop Grid in the Cloud era. We start by introducing Grid computing, legitimately hailed as its forerunner [2]; viewing Desktop Grid systems as a new version based on non-dedicated resources illustrates the Desktop Cloud concept, because both employ non-dedicated resources. The chapter presents a definition of Cloud computing and the characteristics of Cloud systems, whose paradigm is based on technologies discussed here.

The Desktop Cloud is presented as new way to provide services, based on the Cloud business model and using resources of any type. The chapter clarifies the ambiguity by comparing related systems, the Desktop Grid and the Traditional Cloud systems. Challenges and issues are identified, demonstrating why this study focuses on the issue of node failure in Desktop Clouds.

2.1 Grid Computing

Grid computing is a geographically distributed computational platform integrating large-scale, distributed, complex and heterogeneous resources working together to form a virtual super computer. A Grid system can be defined as flexible, secure and coordinated sharing of resources to solve problems in dynamic, multi-institutional virtual organisations [16]. The Grid was motivated by the research community's goal of solving a specific problem via sharing computing resources [17]. Resources in Grids may include clusters, storage systems, databases or scientific instruments [18]. Building dynamic applications that coordinate distributed resources is one of the Grid's strengths, along with utilisation of resources within a particular domain to increase productivity or reduce costs [19] – or both. Ian Foster [20] identifies three essential characteristics of the Grid:

Decentralised manner: computing resources in Grids are managed and coordinated in a decentralised manner. Grids coordinate resources through a multi-institutional virtual organisation.

Standardised protocols: The Grid is a combination of resources from different business domains that interact together using standard, open, general-purpose protocols and interfaces. The standardisation of protocols and interfaces allows Grid participants to establish resource sharing dynamically. The fundamental issues to be considered in this context are authentication, resource discovery and management.

Nontrivial Quality of Service (QoS): The Grid allows access to coordinated resources with respect to various qualities of service such as response time, throughput, availability and security.

2.1.1 Grid Architecture

Grid computing has evolved through three generations [17]: the first shared high-performance computing resources via proprietary solutions; and the second introduced middleware in order to deal with scalability and heterogeneity. Middleware, in the Grid, means the layer that hides the heterogeneity of the underlying infrastructure and provides applications with the necessary environment to run. The third generation aimed to adopt a service-oriented style and web services to accelerate the move towards an e-science infrastructure. Open Grid Services Architecture (OGSA) [21] describes the architecture of a service-oriented Grid environment for both scientific and business usage. In addition, it defines a set of standards, protocols and interfaces to improve interoperability in grid systems.

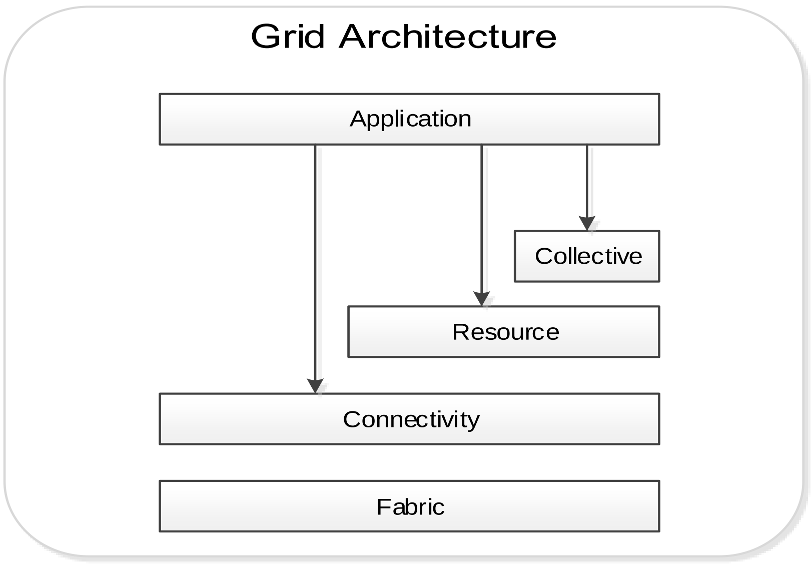

Figure 2-1: Grid Architecture

Figure 2-1 depicts a high-level view of the architecture of Grid computing based on [16]. The fabric layer provides access to shared resources, whether physical or logical. The connectivity layer defines the authentication and communication protocols that enable exchange of data between Grid resources. The resource layer contains protocols to secure negotiation, initiation, control, monitoring, accounting and payment for shared resources.

The collective layer defines protocols and services that enable flexibility in the implementation of different sharing mechanisms. The application layer contains users' applications, which can use services defined on any layer.

2.1.2 Desktop Grids

Desktop Grids are distributed systems that utilise idle resources to perform major computation tasks for scientific projects at low cost [22]. Desktop Grids, depicted in Figure 2-2, share the vision of Grids but use anonymous and untrusted volunteered nodes from the Internet, whereas Grids involve well-known organisations. The aim of Desktop Grid is to harvest a number of idle desktop computers owned by individuals on the edge of the Internet to solve scientific complex problems [23].

Desktop Grid systems can be of two types: public and private systems. In the public Desktop Grid system, people contribute their computers over the Internet by installing a piece of software such as BOINC [24], a famous Desktop Grid platform, to take part in scientific projects. SETI@home [25] is considered a prime example of a public Desktop Grid system. The system relays on resources within an organisation or a group of organisations to take advantage of them when they become idle, mainly with a LAN connection. NotreDame Desktop Grid [26] is an example of a private Desktop Grid system that uses Condor software to oblige staff to join the system when resources are idle [27]. The Desktop Grid proved its success in attracting a huge number of participants; one study shows that SETI@home has employed more than 330,000 computers [5].

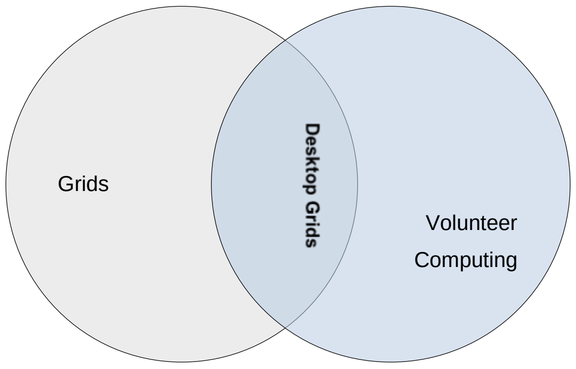

Figure 2-2: Desktop Grids, between Volunteer Computing and Grids

Although the Desktop Grid is considered a special type of Grid computing, there are differences between them, as summarised in Table 2-1, based on [23]. This comparison can illustrate Desktop Grids in terms of advantages and disadvantages. Resources in Desktop Grids are typically non-dedicated computers, while resources in Grids are committed to their system and range from clusters to supercomputers, scientific instruments and so on. The resources in both models are heterogeneous, but more heterogeneous in Desktop Grids because any type of computing resource is accepted. The connection in a Desktop Grid system is quite poor in terms of reliability, with rather limited of bandwidth capability. The connection can be over LAN, in private systems, or the Internet, in public systems. By contrast, the connection in a Grid system is highly reliable with high speeds and throughput.

The poor connectivity in a Desktop Grid system decreases the reliability of nodes. In addition, contributors of computing nodes within a Desktop Grid system can leave the system without prior notification, making expectations of reliability very low compared to Grid systems, according to [28]. Resources in both models are heterogeneous, although they are highly volatile and unreliable in Desktops Grids. The nodes in Desktop Grids are volunteered anonymously by the public; therefore the nodes are untrusted and may include malicious nodes. The types of applications that run on Desktop Grid systems are quite limited compared to Grid systems, restricted to computation-intensive jobs that are independent [18].

Table 2-1: Desktop Grids vs. Grids

	Desktop Grids	**Grids**
Resource	- Anonymous desktops or laptops etc - Non-dedicated resources - Highly heterogeneous	- Cluster, supercomputer, scientific instruments etc - Dedicated - Intermediate heterogeneous
Connection	- Over LAN or Internet - Poor connection - Poor bandwidth	- Dedicated - High speed - High bandwidth
Trust	Anonymous and untrusted	Highly trustworthy providers
Node reliability	Very low	High
Jobs	- Independent - Computation-intensive - High throughput	- Independent or dependant - Computation-intensive *or* data-intensive - High performance

2.2 Cloud Computing

Cloud computing is a new paradigm that promises to deliver computing as a utility, the same as electricity [4]. Cloud computing was driven by giant IT companies such as IBM, Google and others to gain profit from selling computing power. The main motivation for launching Cloud for both CSPs and consumers is cost reduction; the new paradigm can cost just one-fifth of that for electricity, hardware and bandwidth consumption in traditional IT enterprises [29]. Moreover, Cloud users can consume computing power yet pay only for what they use.

The term ‘Cloud computing’ became widely known after IBM and Google announced in 2007 their initiative to participate [30], according to [31]. However, it has been argued that Cloud computing is not new, but rather a new IT paradigm that involves a wide range of existing technologies [32]. There are several Cloud projects in both academia and industry, for example projects from industry such as Amazon EC2,[2] Google App Engine[3] and Microsoft Windows Azure,[4] while the Reservoir model [33] and Nebulas [34] are two academic Cloud projects.

Although there are many Cloud projects in both industry and academia, there is no well-known and accepted definition of Clouds for various reasons. The first is that researchers from different backgrounds, Grid computing for instance, have been involved in Cloud computing [35]. Second, the area is in a state of ambiguity due to confusion about the exact meaning of the term and its capabilities [36]. Third, the Cloud shares the goal of providing computing as utility and achieving high utilisation with other IT paradigms such as Grids and Clustering [4]. Finally, Cloud computing employs various technologies, for instance Web 2.0, that are still evolving [35].

Several papers attempt to present a clear definition of the Cloud such as [4], [37], [36], [38], [39] and [40]. Nonetheless, the definition of Clouds that will be used in this report is the one given by the National Institute of Standards and Technology [40], because this cites its essential characteristics as identified by Vaquero et al. [36]:

> *Cloud computing is an IT paradigm that enables online access to a pool of shared computing resources based on virtualisation technology to allow resources to be rapidly provisioned and scaled up and down according to users’ demands based on paying per usage basis only.*

[2] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/
[3] Google App Engine. http://code.google.com/appengine/
[4] Windows Azure. http://www.microsoft.com/windowsazure/

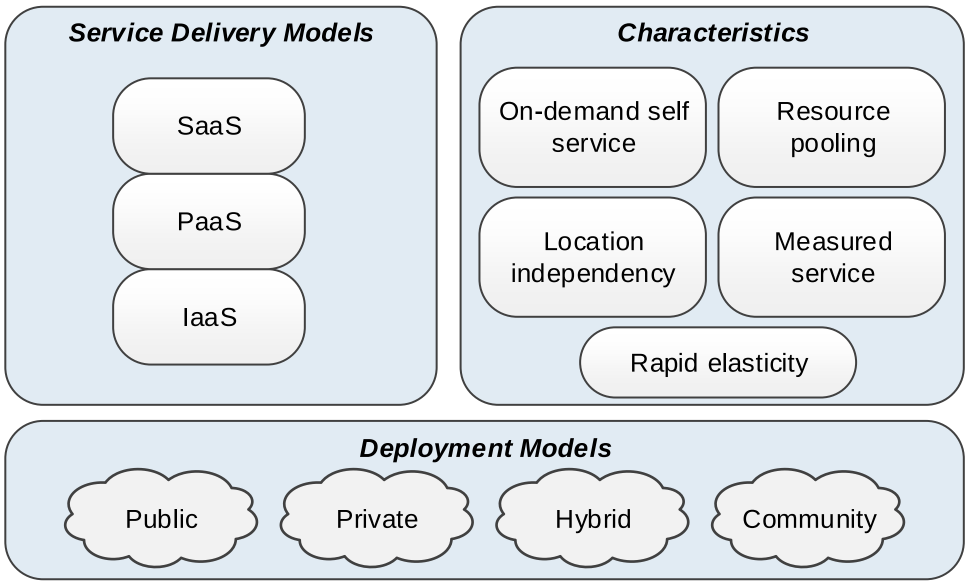

Figure 2-3: Overview of Cloud Structure

Figure 2-3 shows an overview of the structure of Cloud computing, to be explained in the following sections. Typically, the underlying infrastructure of a Cloud consists of one or more data centres, each with a huge number of dedicated computing nodes. This type of Cloud, which harnesses data centre facilities, is called a Traditional Cloud throughout this report.

2.2.1 Characteristics

The characteristics, summarised in Table 2-2, illustrate the meaning of Cloud computing:

- *On-demand self-service*
  Cloud clients can provision computing capabilities in an automatic self-service manner [38], i.e. the consumer can acquire more resources or release them without any human interaction.
- *Resource pooling*
  Cloud providers pool their resources to serve multiple consumers on a multi-tenancy basis [35]. The same PM can host more than one at the same time [41] to reduce operating cost.
- *Online access and Location independency*
  The resources are accessed ubiquitously online using a wide range of devices [40]. Cloud clients have no control over where service providers process these services. However, they may have a choice over the location at a high level of abstraction, for example clients can choose on which continent their data is to be stored on Amazon Cloud.

- *Rapid elasticity and scalability*
  Elasticity is the ability to add or remove computing resources over a short timescale [29]. Scalability means the ability to scale resources up or down, depending on a user's need. These terms are often conflated because both are identical from the perspective of developers on the application layer, yet they are not the same from the perspective of middleware management, the aim of which is to achieve high scalability by employing elasticity through load balancing techniques.
- *Measured Service*
  In Cloud computing, using a server for seven hours costs the same as using seven servers for one hour. Consumers in Cloud pay only for their actual usage. Therefore, CSPs are required to provide accurate service metrics in order to charge users for their usage [32].

Table 2-2: Characteristics of Clouds

Characteristic	**Definition**
On-demand self-service	Gain resource upon request without human interaction [38]
Resource pooling	Assign several VMs to the same PM [35]
Location independency	Location of where services are processed is hidden [40]
Elasticity and scalability	Gain or release Cloud resources in short time upon user's request [29]
Measured services	Cloud capabilities should be measured [32]

2.2.2 Service Delivery Models

There are three common services provided by Clouds: *Software as a Service (SaaS), Platform as a Service (PaaS)* and *Infrastructure as a Service (IaaS)*. SaaS is a delivery model for applications provided by the Cloud, to be run by Cloud users through web tools such as web services. This is the most abstract model of services, where users have no control over the Cloud infrastructure [42]. An example of an SaaS Cloud is salesforce.com.

PaaS offers a platform for developing end-to-end life cycle software development [43] that contains development environment, sets of applications to allow writing code, a set of ready packages to be used by other software and libraries [44]. Google App Engine is a PaaS Cloud. The IaaS delivery model allows users to acquire and release infrastructure resources (e.g. CPU and storage). Virtualisation enables Cloud providers to meet various Cloud consumers' preferences using the same physical resources. Amazon's Elastic Compute Cloud (EC2) is an example of an IaaS model.

2.2.3 Core Technologies

The architecture of Cloud computing comprises four layers: hardware; infrastructure; platforms; and applications layers [45]. These relate to the service delivery models discussed before. Loose coupling is a key feature of Cloud architecture, meaning that each layer can be developed separately from others. The hardware layer is also known as the data centre layer, consisting of all the IT infrastructure required to build Cloud computing. Typically, a CSP has one or more data centres containing all physical resources. Issues relating to this layer are fault tolerance, hardware configuration and traffic management [46]. The virtualisation layer, also known as the infrastructure layer, enables physical resources to be shared, and is based on virtualisation technologies. The platform layer allows developing application on the Cloud (i.e. PaaS). The application layer enables the delivery of applications to end users and, in addition, it supports auto-scaling for applications in order to reduce costs and achieve better performance. Cloud is not a new computing technology, rather a computing model that employs various technologies. This section discusses the core technologies that Cloud relies upon heavily: virtualisation; Service-Oriented Architecture; and web services [38].

Virtualisation is the process of offering a physical computing resource in a virtual fashion to overcome the limitations of the actual resource. It is an essential technology for Clouds that allows the providers to offer computing resources as a utility. The Virtual Machine Monitor (VMM), or hypervisor, isolates the guest OS from the underlying hardware and allows resource multiplexing, that is, running more than one VM on the same PM [42]. However, some researchers argue that virtualisation is a feature that is employed heavily in Cloud computing but it is not an essential feature according [31]. Section 2.2.1 discusses the essential characteristics of Cloud systems illustrating that virtualisation is not necessarily to be incorporated in a Cloud system. Therefore, it can be said that a system can be considered a Cloud system event if virtualisation is not employed.

Technology helps by providing better agility and flexibility [43], as well as reducing costs [41]. Elasticity and resource multiplexing are achieved in Cloud by virtue of virtualisation's maturity [29], allowing better utilisation through resource multiplexing [47]. However, although virtualisation has many advantages for Cloud computing, it impacts on the overall performance of each virtualised, compared to the actual, machine. It is claimed that the overheads for the virtualisation environment ranges from few per cent to about 20 per cent of performance on computation aspects [48]. The reason for this variation is the implementation of the VMM as well as whether the guest OS is aware of being run on a VM

[41]. In addition, virtualisation introduces new security threats in Cloud computing. For example, resource multiplexing can allow an attacker to acquire information about a target VM on the same PM, using side-channel attack [49].

Service-Oriented Architecture (SOA) is an architectural approach to delivering a reusable IT service in a higher-level abstraction, independent of its underlying infrastructure [50]. However, SOA is not a technology but a guide to business solutions to create computing components in such a way as to allow them to be easily reused and extended. It relies on WSDL (Web Service Description Language) and SOAP (Simple Object Access Protocol) to standardise interaction between different components [51]. This mechanism allows third parties (Cloud brokers) to resell these services to end users. Thanks to SOA and virtualisation, the underlying infrastructure is abstracted without revealing much to users [31].

Web services may be seen as a way of implementing components on SOA. A CSP exposes Cloud services as web services on an SOA basis [35]. Both WSDL and the REST protocol are used to describe Cloud API, since they are widely used to describe web services [42]. Service orchestration is defined as the automated coordination and management of different services, and Cloud computing relies on it to coordinate and deliver scalable and self-automated services from other resources. Web-oriented architecture that defines the interaction between different web applications is a key concept in Web 2.0; the idea behind it is to enhance creativity and to improve collaboration on the web.

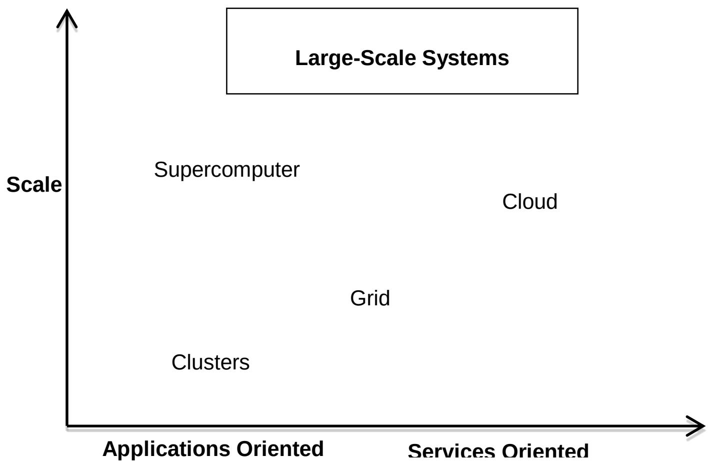

Figure 2-4: Cloud and Service Oriented

Figure 2-4 summarises, based on [2], where Clouds stand in terms of applications and service orientations. Virtualisation and SOA enable Cloud providers to abstract their systems and thus to isolate consumers from the underlying infrastructure. The services in Cloud computing are web services that interact and are managed according to the service orchestration approach.

2.2.4 Deployment Models

There are four types of Cloud deployment models, also known as Cloud types: public; private; community; and hybrid Clouds.

- *Public Cloud* is the popular type in which a Cloud provider offers its services. Amazon, Google App Engine and Windows Azure are examples [43].
- *Private Cloud* is the type where a Cloud provider offers services exclusively to a single organisation, also known as internal Clouds. Some providers use their own private Clouds to test services before offering them to the public [46]. There is various open source middleware developed to manage private Cloud, Eucalyptus [12] for example.
- *Community Clouds* are formed by a group of organisations with common interests to use local infrastructure resources [45]. However, the community Cloud is based on the idea of exploiting local infrastructure to offer free or low cost Cloud services. Nebulas [34] and Cloud@home [52] are classified as community Clouds.
- *Hybrid Clouds* combine two or more Clouds types to achieve maximum cost reduction with maximum utilisation. For example, a private Cloud may be used within an organisation to process sensitive data, while exploiting public Clouds for cost reduction.

2.2.5 Cloud Actors

There are five Cloud actors, according to the Cloud Computing Reference Architecture proposed in [53], as depicted in Figure 2-5:

- *Provider*: responsible for the management and administration of the physical infrastructure of Clouds. The management role includes: maintenance of PMs; cooling systems; and VM allocations and resource management. A Cloud provider is also known as a CSP.
- *Auditor*: ensures and verifies that standards are met within Cloud systems. The Cloud auditor can evaluate the services of a CSP in terms of security controls, privacy impact, performance and so on.
- *Consumer*: a person or organisation who exploits services offered by CSPs.

- *Broker*: the middleman between a CSP and end users. End users can acquire services directly from Cloud providers, but it can be complicated. A broker can help end users to aggregate services they need effectively.
- *Carrier*: acts as a means of connecting consumers, brokers and providers in a Cloud environment.

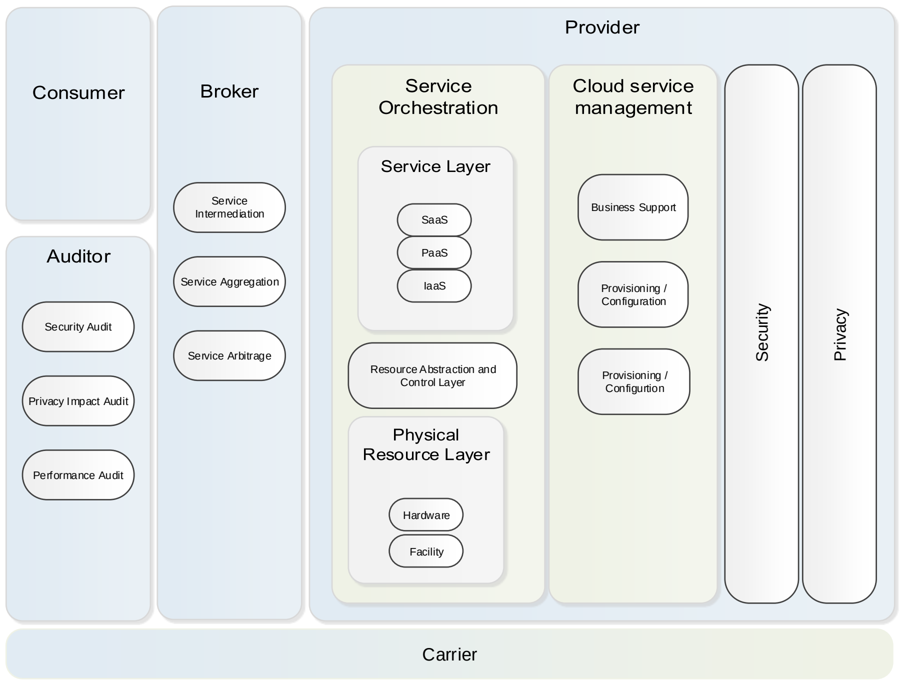

Figure 2-5: NIST Cloud Computing Model

2.3 Desktop Clouds

Desktop Cloud computing is the idea of benefiting from the computing resources around us to build a Cloud system that achieves better usage of resources than having them lying idle. The success of Desktop Grids stimulated the idea of harnessing idle resources to build Desktop Clouds, and the term Desktop derives from Desktop Grids, since both are based on Desktop PCs, laptops and so on. Similarly, the term Cloud derives from Desktop Clouds, providing services based on the Cloud business model.

Several synonyms for Desktop Clouds have been used in the literature, such as Ad-hoc Clouds [54], Volunteer Clouds [11], Non-Dedicated Clouds [55] and P2P Clouds [56]. The literature shows that little work has been undertaken in this direction. The Ad-hoc Cloud is the idea of harvesting distributed resources within an organisation to form a Cloud [54]. Nebula [9],[34] is a project aiming to exploit distributed resources in order to create a Volunteer Cloud offering services free of charge. Cloud@home [40],[47] is a project representing the @home philosophy of Cloud computing and its goal is to form a new model of Cloud computing to which individual users contribute over the Internet. In addition, Cern has announced an initiative to move their Desktop Grid project, called LHC@home, onto the Cloud [58]. Moreover, the authors in [55] suggest that non-dedicated resources can be exploited by Cloud providers in the event that their local infrastructure cannot meet consumer requests at peak times.

Desktop Clouds have some advantages over Traditional Clouds. First, Traditional Clouds have a deleterious impact on the environment since their data centres consume massive amounts of electricity [7]. The second advantage is the cost effectiveness of Desktop Clouds for both Cloud services providers and their consumers; the former have no need to build new data centres to meet future increasing demand, while the latter receives their services at lower prices than from Traditional Clouds, if not free. Desktop Clouds help to reduce energy consumption since they use already-running, undedicated resources that would otherwise lie idle. Some studies show that the average percentage of local resources lying idle within an organisation is about 80% [8]. Furthermore, Traditional Clouds are formed from a limited number of data centres located around the globe. Therefore, they are inefficient in terms of data mobility and pay little attention to clients' location [9]. Finally, Traditional Clouds are centralised, with the potential for a single point of failure issue should a CSP go out of business. By contrast, Desktop Clouds manage and offer services in a decentralised manner.

2.3.1 Scenario

This section depicts two scenarios in forming Desktop Cloud systems: one is private and another public. Suppose a group of universities wishes to benefit from its computing resources by forming a Cloud. The resources range from PCs to servers, each called a Cloud node, and a node can join the Cloud when it becomes idle. This scenario is motivated by Condor [27], middleware for the Desktop Grid with the aim of recycling idle CPU capability to solve large scientific problems via a private Desktop Grid system. Users in a Desktop Cloud submit their requests to acquire services with their requirements, as stated in the service level agreement between a client and the Cloud interface, and these are processed in the virtualisation layer lying above the Cloud's physical nodes. Virtualisation isolates the guest operating system from the physical host machine, improving security and preventing unauthorised access by the two parties [59]. This system is considered a private Desktop Cloud system.

The other scenario is a public Desktop Cloud that allows people to contribute their own computing resources to be used by Cloud clients [60]. This type of system is motivated by projects such as SETI@home [25]. In this scenario, PMs are collected from people contributing their computing resources online to join a public Desktop Cloud system to serve a particular purpose for a period of time. For example, the purpose may be to serve the research field by allowing researchers take advantage of PMs to solve research problems requiring a massive number of computers.

There are two main differences between private and public systems. The first is that resources in private systems are limited to system owners, while resources in public systems are contributed by the public, so the number of resources in public systems is expected to be higher. Second, resources in private systems are mainly connected via LAN connections while in public systems they are connected via WAN, so the number of failure events in public Desktop Cloud systems may be expected to be higher.

Table 2-3: Essential Features of Cloud Systems

Feature	**Traditional Cloud**	**Desktop Cloud**
Elasticity	√	√
On-demand self-service	√	√
Virtualisation	√	√
Service delivery model	√	√
Ease of use	√	√

2.3.2 Desktop Clouds vs. Traditional Clouds

This section clarifies Desktop Clouds further by comparing them with Traditional Clouds to illustrate the similarities and differences. Both possess the same essential features of Cloud computing [36], as in Table 2-3. The features are elasticity, on-demand self-service, virtualisation, a service delivery model and ease of use. Both of these Cloud types share the elasticity feature, meaning that a Cloud's user can acquire computing services and scale up or down according to their need in a short time. The services are online self-service, which means that Cloud services can be acquired and released in an automatic way. Both employ virtualisation to separate VMs from PMs. The model of services delivered by Cloud systems should implement the 'pay per use' base, meaning that Cloud services are delivered on a time-based contract between a CSP and customers. Users of Desktop Clouds are not expected to pay for their usage, but can achieve services on the same principle. For example, a Desktop Cloud's user can gain a couple of VMs, for example for three hours, similar to what happens in Traditional Clouds. The 'ease of use' principle means that clients can use a specific service without the need to make many changes to their work. Both Traditional Clouds and Desktop Clouds let their users harness services without making significant changes to their code.

Table 2-4: Traditional Clouds vs. Desktop Clouds

Feature	Traditional Clouds	Desktop Clouds
Resources	Dedicated	Non-dedicated and volatile
Cost	Relatively high	Cheap
Location	Limited	Distributed
Services	Reliable and available	Low availability and unreliable
Heterogeneity	Heterogeneous	Very heterogeneous

However, there are several differences, as depicted in Table 2-4. The first is in the resource layer; the Desktop Cloud is made of resources that are non-dedicated, that is, they are not made as part of the Cloud infrastructure. The physical nodes in Desktop Clouds are expected to be highly volatile, due to the fact that they can fail without prior warning. The failure of nodes can be the result of machines crashing, connectivity problems or the owner of a PM decides to leave the Cloud. High volatility of resources can have a negative impact on availability and performance [11]. By contrast, the infrastructure of Traditional Clouds consists of a large number of computing resources located in data centres made to serve clients in the Cloud. Moreover, resources in Desktop Clouds can be distributed around the globe, so anyone can contribute from anywhere, while resources in Traditional Clouds are limited to the location of the data centre hosting them. Although resources in both Desktop Clouds and Traditional Clouds are both heterogeneous, they are more so in Desktop Clouds.

2.3.3 Desktop Clouds vs. Desktop Grids

The vision of providing computing services as utility services is shared by Grids and Clouds. In addition, many researchers assert that Cloud computing evolved from Grid computing [2]. This has caused some confusion between Grids and Clouds and several papers have been published that compare and contrast Grids with Clouds, such as [2], [37], [36] and [61]. This comparison helps to understand the differences.

Table 2-5 summarises the similarities and differences: first, they both rely on dedicated resources that are highly reliable. In fact, some of the Grid infrastructure can be used as parts for Cloud infrastructure. In addition, Clouds implement virtualisation to abstract PMs. Virtualisation enables resource multiplexing, meaning that more than one VM can be assigned to the same physical node. Clouds offer services on a ‘utility basis’, which means that users gain services and pay only for their actual usage, whereas the Grids business model is based on a ‘project oriented’ model, whereby every client is assigned a certain time to use a particular service, whether or not the client actually uses it [2]. As mentioned before, Grid computing was devised to serve and solve some research problems in research communities, so it is quite normal to have a single-purpose Grid project. For example, neuGrid is a Grid project that enables neuroscientists to carry out research regarding degenerative brain diseases [62]. Cloud computing, on the other hand, offers services to a wide range of customers for various purposes. In addition, both Grids and Clouds can achieve QoS guarantees, but at different levels. For the former, at the application level users can implement some sort of mechanisms to ensure QoS, while for the latter it is granted by CSPs.

Table 2-5: Grids vs. Traditional Clouds

Feature	Grids	Traditional Clouds
Resources	Dedicated and reliable	Dedicated and reliable
Resource multiplexing	Not employed	Employed
Business model	Project oriented	Pay per use
Application domain	Research projects	Various purposes
Quality of service	Guaranteed (at application level)	Guaranteed by CSPs

Desktop Clouds can be conflated with Desktop Grids in a similar way to Clouds and Grids. Table 2-6 shows a summary of the comparison of Desktop Clouds with Traditional Clouds on one side, and Grids and Traditional Clouds on the other. The resources have similar features, being non-dedicated and unreliable in both Desktop Grids and Desktop Clouds.

Table 2-6: Desktop Grids vs. Desktop Clouds

Feature	Desktop Grids	Desktop Clouds
Resources	Non-dedicated	Non-dedicated
Infrastructure	Unreliable	Unreliable
Resource sharing	Not supported	Supported
Business model	Project oriented	Cloud model
Application domain	Research projects	Generic purposes

In fact, a Desktop Cloud can be built on top of a Desktop Grid infrastructure. However, Desktop Clouds employ virtualisation that enables resource sharing between users via resource multiplexing. Finally, Desktop Clouds follow the business model offered by Traditional Clouds in terms of scalability and elasticity. Users in Desktop Clouds can acquire VMs as they desire, and scale them up or down according to their needs. Desktop Grids employ the ‘project oriented’ concept that is used by Grids. The Desktop Grids’ main application domain is research projects, because researchers use idle resources to help them solve complex research problems. This goal is shared with Desktop Clouds, but can be more generic. For example, Desktop Clouds can be used by CSPs in Traditional Clouds when a CSP owned data centre cannot meet user demand. Thus, Desktop Clouds can have a more generic purpose than Desktop Grids, by virtue of the business model that enables them to deal with several clients.

Table 2-7: Desktop Clouds vs. Large-Scale Systems

	Desktop Cloud	Traditional Cloud	Grid	Desktop Grid
Elasticity	√	√	X	X
Virtualisation	√	√	X	X
Idle resources	√	X	X	√
Ease of use	√	√	X	X

Table 2-7 shows a summary of comparisons between Desktop Clouds and Traditional Clouds, and between Grids and Desktop Grids. Both Clouds rely heavily on virtualisation. In addition, both let their users harness services without making significant changes to their code. However, this is not the case in Grids and Desktop Grids, where users are expected to know in depth the middleware used in order to harness the services offered [2]. Both Desktop Clouds and Desktop Grids depend on computing resources when they become idle.

2.3.4 Research Challenges

This section describes several research issues that need further attention in Desktop Clouds. Some of these challenges are inherited from Cloud computing, while others are driven by the highly volatile resources employed, a legacy of Desktop Grids. Table 2-8: Research Challenges summarises the issues of Grids, Desktop Grids, Cloud and Desktop Clouds; Desktop Clouds inherit the challenges and issues of these systems.

Table 2-8: Research Challenges

Computing Model	Challenges	References
Grids	Interoperability	[16][63]
	Quality of service	[64][19][17]
	Security & authentication	[65][17]
	Scalability	[17][19]
	Resource management	[19][16]
Desktop Grids	Resource management	[66][67]
	Nodes availability	[22][68]
	Performance prediction & metrics	[66][22]
	Resource volatility	[23][68][69]
Traditional Clouds	Security & trust	[49][70][71]
	Migration to Cloud	[29][72]
	Interoperability & portability	[32][43][73]
	Resource management	[43][32][46]
	Pricing models	[74][32][75]
	Availability	[29][76][77]
Desktop Clouds	Security	[55][11]
	QoS	[34][78]
	Resource contribution	[11]
	Node volatility	[79][55][11]
	Resource management	[52][55]

2.3.4.1 Security

Security is a major concern preventing organisations from moving onto the Cloud [42]. Ristenpart et al. show that an attacker can uncover the actual location of a particular VM [49], and a cross-VM side channel attack can reveal critical information about the targeted VM by placing a malicious VM on the same PM. Further concerns arise in Desktop Clouds, as both consumers and contributors are from the public, so security can be a major issue in this context. In addition to the threats previously mentioned for the Cloud, both consumers and contributors themselves take on risk when they join a Desktop Cloud. A contributor can put his own data at risk by allowing access to a virtual image located in his machine. Likewise, consumers are vulnerable to malicious contributors; nodes in Desktop Clouds are more vulnerable to outside attack due to weaknesses in local antivirus software and firewalls.

Virtualisation can be vital in order to isolate the host completely from guest operating systems and, thus, prevent any unwanted access from either party. Trust mechanisms can be employed in this matter. For example, a Desktop Cloud can maintain a behaviour table which contains information about both consumers and contributors. The table can be used to decide which parties are trustworthy enough to join the cloud. Furthermore, Desktop Clouds should rely on autonomous mechanisms such as sandbox or certification in order to prevent various attacks from participants [80].

2.3.4.2 Quality of Service

Desktop Cloud systems are only expected to offer services at a low level of reliability and availability, due to the fact that they depend on unreliable volunteered resources that can join or leave the system without prior warning for various reasons [55], so the quality of service can be affected hugely [34]. Traditionally, CSPs provide services to end users based on a Service level agreement (SLA) contract between them to ensure metrics of quality of services are met.

The authors in [78] propose C@H as the QoS management approach to ensure that it is guaranteed in Desktop Cloud systems. However, there is no actual evidence provided to support their claim. For example, the availability of the service should be at least up to a certain level, and the availability of individual nodes is considered a primary issue [11]. It is estimated that in volunteer projects resource non-availability can reach 50% [22], yet the availability of each individual node can affect the service quality of Desktop Cloud system. Andrzejak et al. propose a technique to predict the availability of a group of high volatility resources [81]. This prediction can help to estimate when a PM will leave the system but does not, however, provide any guarantee that the quality of service is meets the SLA.

2.3.4.3 Resource Contribution

Another issue to be considered in Desktop Clouds is how to motivate people to become involved in Desktop Cloud systems; the research is mainly into how to motivate people to contribute their computing resources to be part of a Desktop Cloud system [11]. There are obvious reasons why people wish to contribute their computing resources to research projects such as SETI@home, for instance for the sake of improving knowledge. It may be said that convincing people to contribute to Desktop Grids is far easier than getting them to contribute to Desktop Clouds. A contributor needs only to install a piece of software, BOINC software for instance, on a SETI@home system to run small batches of jobs, but it is more complicated to be a part of a Desktop Cloud, as people need to install virtualisation software. Some would be reluctant to do so, especially those with no experience in computer science.

A study might be carried out to obtain the public's opinion on what would make people join a Desktop Cloud system. This survey would help to develop approaches to stimulate joining them, for instance by introducing a credit scheme for contributors: the more a contributor offers their available resources in a Desktop Cloud system, the more credit they accrue to secure benefits from the system; for example, credit might be spent by the contributor on high priority access to services.

2.3.4.4 Node Failures

One of the main issues in Desktop Clouds is the high rate of node failure during run time [82]. Desktop Clouds are only expected to offer services at a low level of reliability as they depend on unreliable computing resources that can join or leave the Cloud without prior warning, yet this increases the risk of node failure [55]. In Cloud computing, VM instances are hosted and run in PMs so, if a PM fails, all VM instances allocated to it are destroyed. Consequently, PM failure means all the tasks running on the VM instance are lost.

The problem is crucial in Desktop Clouds as the number of node failure events can be high, including events that cause the node to leave the Cloud for any reason. PMs can leave without warning if they become busy with local tasks, moreover they are connected to the system via unreliable connections, further increasing the risk of failure. Devising a fault tolerance mechanism is essential, and to increase the efficiency of Desktop Clouds this challenge demands attention. Accordingly, this research focuses on solving the issue of node failure by designing a novel mechanism that ensures that, in the presence of node failure, jobs submitted to a Desktop Cloud system are completed.

2.3.4.5 Resource management

Resource management is an open field for research in Desktop Clouds [54]. It can play an important role in improving the performance of Desktop Clouds [78]. Resources in Desktop Clouds are highly heterogeneous, so managing them is considered problematic. Virtualisation plays a key role in Desktop Clouds because it virtualises contributed resources and delivers them to users as VMs. Desktop Clouds face the challenge of developing a VM allocation mechanism that is able to: a) manage non-dedicated, heterogeneous resources; b) deliver a virtualised machine to the upper layer in Desktop Clouds; and c) provide a fault tolerance policy that can deal with the issue of high levels of node failure.

It has been pointed out that the absence of central management in Desktop Clouds causes a major issue in terms of reliability and state maintenance in the event of failure [82]. The infrastructure of Desktop Cloud contains nodes that are highly volatile. Therefore, fault recovery mechanisms are crucial in order to improve reliability in this environment [11]. In addition, Desktop Clouds require a means of interacting with other Clouds for data migration or to gain extra computing resources [52]. However, the authors in [83] point out that none of the available open-source Cloud management middleware provide a resilience feature for Cloud systems.

2.3.5 Desktop Cloud Architecture

According to the authors in [84], the architecture of a Desktop Cloud system is divided into four layers: (i) the service layer is a front-end between Desktop Cloud and users; (ii) the virtual layer is responsible for provisioning VMs; (iii) the physical layer is responsible for managing resources found in the resource layer; and (iv) the resource layer contains resources that form the underlying infrastructure. Each computing node joining the Cloud is called a Desktop Cloud node. Users in the framework can be individual consumers from any domain.

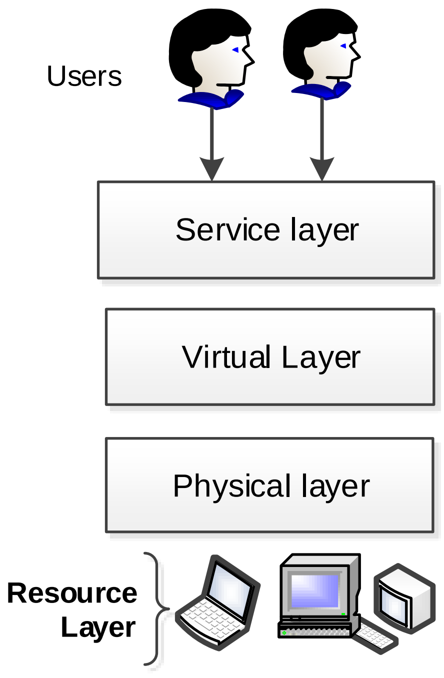

Figure 2-6: Abstract of Desktop Cloud Framework

2.3.5.1 Service Layer

The service layer, Figure 2-7, provides IaaS services via an interface to customers based on an SOA approach [50]. The business model in Desktop Clouds is similar to that of Traditional Clouds, aiming to provide scalable computing services to users. The layer accepts various tasks from Cloud users. Users should be authenticated in the ‘authentication’ entity before they can be granted access to services. The tasks are interpreted in the ‘task analyser’ to decide whether to accept them or not, given the available resources. The 'scheduler' is responsible for implementing the scheduling policy, assigning a task to a VM [85]. VMs are provided by the virtual layer.

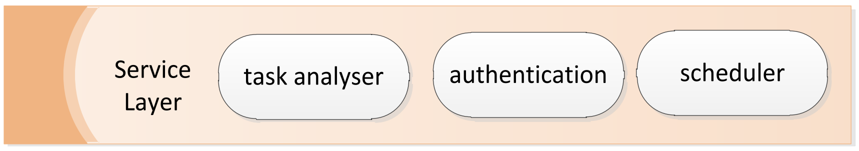

Figure 2-7: Service Layer

2.3.5.2 Virtual Layer

The virtual layer, Figure 2-8, is responsible for provisioning, managing and controlling VMs. In the IaaS Cloud, users can gain computing resource in virtualised forms, involving the creation of a virtual layer between users and PMs. The 'VM manager' component provides VMs to Cloud users upon request. The first step to take is virtual placement, allocating a virtual machine to a PM. A VM placement mechanism must take into consideration that the infrastructure of Desktop Clouds consists of highly diverse resources. The VM manager can assign more than one VM to the same PM (VM consolidation) in order to preserve power and improve resource utilisation. The authors in [86] show that VM consolidation can help in reducing the power consumed by physical resources. The 'VM monitor' observes the behaviour of VMs used by users to ensure the SLA is not violated. The security of VMs is crucial. The security component works to protect both the host and guest operating systems from unauthorised access, as virtual technology isolates the guest operating from the host system [87].

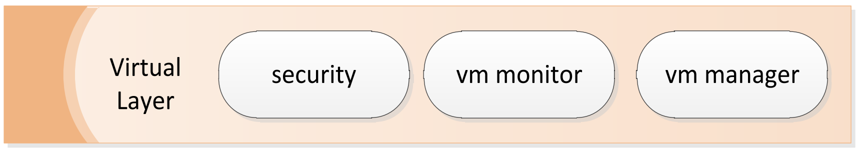

Figure 2-8: Virtual Layer

2.3.5.3 Physical Layer

The physical layer considers managing computing nodes that form the infrastructure of Desktop Clouds. The layer contains three components, interacting with each other as shown in Figure 2-9. The 'discovery' keeps track of all available resources in the Cloud. The 'aggregator' decides which resources are preferable to serve users' requests. For example, a ranking table can be used to classify various resources according to their behaviour in the past to decide which is more reliable [55]. The 'allocator' is in charge of allotting VMs to physical resources. It can implement a VM allocation mechanism with the aim of providing better performance. The component should be able to cope with a high resource fault rate by employing suitable fault tolerance mechanisms.

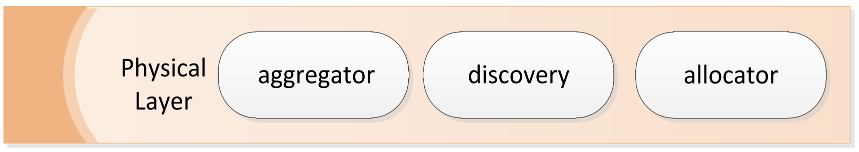

Figure 2-9: Physical Layer

2.4 Summary

This chapter presented an overview of Cloud computing and related computing models: Grid and Desktop Grid. It showed that there is no consensus on what ‘the Cloud’ actually means, so outlining the essential features of Clouds is crucial to understanding what is Cloud computing. Computing resources in Cloud computing are available online upon request. Elasticity is a crucial feature, meaning that users can scale resources up in a short time yet still only pay for what they consume.

This chapter presented the Desktop Cloud as a new type of Cloud computing. Its infrastructure is made up of non-dedicated idle resources. Distinctive characteristics of Desktop Clouds are: low cost, non-dedicated and heterogeneous resources, and the low quality of services. Several research issues were discussed in the area of Desktop Cloud systems. Node failure is identified as being one of the major obstacles to the advance of Desktop Cloud systems, and can be solved by implementing a fault tolerance mechanism that ensures an acceptable level of service without interruption.

The next chapter reviews related works developed in the area of Cloud computing in the literature that can be used to tackle the issue of node failure in Desktop Cloud systems.

Chapter 3: Literature Review

The previous chapter introduced Desktop Clouds, a new type of Cloud that aims to harness computing PMs to provide Cloud services when idle, and identified the research challenges to be tackled. This study focuses on how to design a VM allocation mechanism to improve throughput in Desktop Cloud systems in the presence of node failure.

This chapter starts by explaining the meanings given to the term VM allocation mechanism in the literature, as there is no consistent definition. This is because the mechanism may contain several techniques such as VM placement and VM migration. This leads researchers to adopt different terms, as this chapter describes. The next section proposes a taxonomy and survey of the various VM allocation approaches to the VM allocation problem. They can be divided into five types: heuristic; power aware; performance aware; network aware; and failure aware.

3.1 VM Allocation Mechanism

There are various synonyms for and interpretations of the VM allocation mechanism in Cloud computing, because the paradigm is relatively new and there is no consensus about either the Cloud or its components [36]. This section explains the meaning of a VM mechanism and shows how related areas of research are confused with VM allocation, although outside its scope.

3.1.1 Definitions

A VM allocation mechanism can be defined as any technique or policy to manage the placement of VMs to PMs in Cloud systems. It involves the option of migrating a VM or group of VMs from the hosted PM to another with the aim of optimising Cloud systems by reducing energy, reducing power consumption or enhancing performance throughput. There are several processes involved in the VM allocation mechanism, namely VM Provisioning, PM selection, VM consolidation, VM monitoring and VM migration.

Van et al., define *VM provisioning* as the stage of providing VM instances to Cloud users or brokers in order to process tasks [10]. A VM instance is one of a number of VM classes, each assigned a certain CPU power and RAM, for users to select.

PM selection is the process of selecting a suitable PM to host a VM instance. It is also termed ‘resource selection’, as in [82]. *VM placement* is the step of actually instantiating and allocating a VM instance to the PM chosen at the PM selection step [88]. *VM consolidation* is the process of allocating more than a single VM instance to a PM, made possible in Cloud computing by virtualisation technology [89]. Its aim is to improve resource utilisation by assigning as many VMs as possible to the same PM. A synonym for VM consolidation is ‘server consolidation’, as used in [90], or ‘VM multiplexing’, as in [91].

VM monitoring is a module to check currently running VM instances in order to optimise the Cloud system by triggering *VM migration* to prevent degradation of the VM performance due to resource overload, to improve the quality of service or to reduce power consumption by improving resource utilisation [92]. *VM migration* is the process of migrating a VM instance from its hosted PM to another to achieve various objectives such as reducing power consumption or improving performance in processing Cloud applications [93]. Thanks to virtualisation technology, a VM instance can be migrated either live or non-live from one PM to another. Live migration refers to moving a VM instance during run time without the need to pause, so the process of VM migration can be conducted with minimum impact on VM performance [94].

3.1.2 Scope

There is confusion and misunderstanding about the definition of the VM allocation mechanism. For example, some researchers term it the ‘VM provisioning mechanism’, as do Meng et al., [91], because its main goal is to provision VM instances to end users on request. ‘VM management’ is another term for the VM allocation mechanism, as in [10], referring to the process of allocation and migration of VM instances. Although the term is attractive to describe the steps required to instantiate VMs, it might be confused with the role of VM monitor on VM management platforms such as KVM [95]. In addition, Verma et al., term it ‘VM placement’, referring to a phase within the VM allocation mechanism [96]. Similarly, the term ‘VM mapping mechanism’ is used by Calheiros et al., to describe the VM allocation mechanism, although it is a step within the mechanism [97]. The term ‘VM resource allocation’ is used in the literature in work by [98], for example. Similarly, the term ‘VM allocation policy’ is used by [89] to refer to the mechanism and ‘VM placement policy’ by [99]. However, this researcher considers that these studies, and similar, all fall into the area of VM allocation mechanisms.

This study focuses on designing a mechanism of several stages to ensure an acceptable level of successfully executed tasks in the presence of node failure. Therefore, its scope is limited to the perspective of CSPs. Other research areas can easily be confused with this field, because the term ‘Cloud’ involves the study of many dimensions by researchers of varying backgrounds. For example, the study of reducing running costs for end users in Cloud computing is sometimes called ‘resource allocation in the Clouds’, as mentioned by [100]. Their work concerns the reduction in the Cloud of the number of requested VM instances by end users, in order to cut their costs, since fewer running VMs and shorter duration of usage are major factors of charges. However, such works are considered beyond the scope of this study, which is concerned only with the techniques, mechanisms and approaches to help CSPs to improve performance, reduce power consumption … etc. This is the technical side of Cloud systems.

Another research area that is beyond the scope of the VM allocation problem is task scheduling techniques. These are employed by CSPs or Cloud brokers to distribute Cloud tasks to a list of available VM instances [101], for instance the work that has been done by [102]. This research area is easily confused with the VM allocation area because they are both typical research issues in Cloud computing and both may have the same goal, such as reducing running costs, at certain times. Another research area that is similar but beyond the scope of this study is the VM spot market, the notion of CSPs offering VM instances at a lower price than usual. Researchers try to design and develop mechanisms, often called resource allocation policies or mechanisms, as in [103], that let Cloud users specify their requirements and desired budget using VM instances. The mechanisms then identify, negotiate and locate VM instances to fit these requests.

3.2 Taxonomy of VM Allocation Mechanisms

The previous section defined the VM allocation mechanism and the steps involved, while the previous chapter explained that developing VM allocation mechanisms for Cloud systems is an open area for further research. This subsection proposes a taxonomy of earlier studies tackling the problem of VM allocation. This divides the studies into five types, according to their purpose.

3.2.1 Heuristic Approaches

In Cloud computing, the problem of VM allocation mechanism can be formulated as follows. The set of VMs and the set of PMs may be seen as a version of a Bin Packing problem [104]; in its classic form this means packing objects of different volumes into the least possible number of bins [105]. The problem is how to place multiple VMs into the fewest PMs to achieve better utilisation and to minimise power consumption. However, the problem needs further attention because it involves another vector. If it exceeds a certain level, the number of VMs placed in each PM can cause performance degradation in VMs [106], resulting in more SLA violations. It can also lead to PMs consuming more energy than normal due to a state of over-utilisation [89]. Therefore, the Bin Packing problem has to be extended to involve the allocation of as many VMs as possible to the same PM with two restrictions: non-violation of SLA and non-excessive power consumption due to over-utilisation [88].

Several Bin Packing heuristic solutions for VM placement were evaluated by [98], who investigated the First Fit (FF), Least Fit First (LF) and Next Fit (NF) algorithms for node-selection decisions in Traditional Clouds. Their experiments showed that the LF algorithm is slightly better in terms of usage. Similar work was undertaken to evaluate the FF, NF and Best Fit (BF) algorithms with regard to performance by [107], who concluded that the BF algorithm performs best. However, the heuristic algorithms cannot guarantee better results, as aforementioned studies showed, therefore VM allocation policies need to define several steps to achieve better results. These heuristic approaches are criticised for being unable to guarantee the optimal solution, as stated by [10].

Such heuristic solutions can be applied to PM selection only for VM placement or VM migration steps, although the VM allocation mechanism should also be responsible for monitoring running VM instances in order to migrate VM instances during run time in order to optimise Cloud systems, according to [108], [104], [98] and [82]. The FCFS, Greedy and RoundRobin VM allocation mechanisms are implemented in open source resource management for Cloud computing such as Eucalyptus, OpenNebula and Nimbus according to [108] and [109]. These offer VM placement and may also be used to optimise Cloud systems by reducing power consumption or improving performance. However, Eucalyptus and Nimbus only employ static VM placement using FCFS or Greedy mechanisms in order to select the proper PM to host a new VM instance, thus incurring no disadvantage, according to [109].

The FCFS mechanism [110] allocates a VM to the first available PM that can accommodate it. In Cloud systems, the FCFS mechanism employs the FF heuristic solution for VM placement, while Eucalyptus uses it to place a VM instance with the first available PM able to accommodate it [111]. The FCFS mechanism cannot be used to serve any allocation objectives such as improving resource utilisation or bandwidth optimisation.

The Greedy mechanism, based on the Greedy algorithm [112], allocates a VM to the PM with the highest level of utilisation. In this context the Greedy mechanism employs the BF solution, which is the PM with highest utilisation level in a pool of PMs. If the chosen PM cannot accommodate the new VM, then the next most utilised PM will be allocated. Nimbus employs the Greedy mechanism to place a new VM instance at the PM with highest utilisation level, if able to do so to improve utilisation of Clouds' PMs; however, it does not employ any VM migration steps to migrate the VM instances during run time. It can be effective during runtime by optimising running PMs, migrating VMs in PMs with low levels of utilisation to more heavily utilised PMs, however, it can also lead to over-utilised PMs, adversely affecting performance in running VMs, according to [89].

RoundRobin, based on the RoundRobin scheduling algorithm [113], is a VM allocation mechanism that allocates a set of VMs to each available physical host on a rotating basis with the aim of distributing PMs' loads. It employs the LF heuristic solution, so selects the PM with the least used resources (CPU and RAM) to host VM instances for initial placement or migration. The RoundRobin mechanism can be used during runtime to migrate VM instances from one PM to another just to ensure that the load is distributed among Clouds' PMs; however, it may be argued that it leads PMs to consume more energy, because it does not pay attention to improving resource utilisation and results in many under-utilised PMs.

3.2.2 Power-Aware Approaches

The literature shows that the focus is on how to minimise the power consumed by physical nodes in order to maximise revenue for CSPs. Researchers are motivated to tackle the issue because power in data centres accounts for a large proportion of maintenance costs [114]. The idea is that better utilisation leads to more servers that are idle, so can be switched to power saving mode (e.g. sleep, hibernation) to reduce their energy consumption. According to Kusic et al., an idle machine uses as much as 70% of the total power consumed when it is fully utilised [13].

Srikantaiah et al., studied the relationship between energy consumption, resource utilisation and performance in resource consolidation in Traditional Clouds [106]. The researchers investigated the impact of resource high utilisation on performance degradation when various VMs are consolidated at the same physical node, introducing the notion of optimal points. They argued that there is a utilisation point that allows placement of several VMs at the same physical node without affecting performance. Once this point is reached in a PM, no new VMs are placed, and the proposal is to calculate this optimal point of utilisation then to employ a heuristic algorithm for VM placement, since the authors defined the consolidation problem as a multi-dimensional Bin Packing problem and showed that the consumption of power per transaction results in a ‘U’-shaped curve. They found that CPU utilisation at 70% was the optimal point in their experiment, but that it varied according to the specification of the PMs and workload. The approach is criticised because the technique adopted depends heavily on the type of the workload and the nature of the targeted machines [115].

Verma et al., presented ‘pMapper’, a power-aware framework for VM placement and migration in virtualised systems, where the monitoring engine collects current performance and power status for VMs and PMs in case migration is required [96]. The allocation policy in pMapper employs *mPP,* an algorithm that places VMs on servers with the aim of reducing the power they consume. The algorithm has two phases. The first is to determine a target utilisation point for each available server based on their power model. The second is to employ a First Fit Decreasing (FFD) heuristic solution to place VMs on servers with regard to the utilisation point of each. The optimisation in the framework considers reducing the cost of VM migration from one server to another. The migration cost is calculated by a migration manager for each candidate PM in order to determine which node is chosen. The work is criticised as it does not strictly comply with SLA requirements [86]; the proposed allocation policy deals with static VM allocation where specifications of VMs remain unchanged. This is not the case in Cloud computing, where clients can scale up or down dynamically. In addition, it requires prior knowledge of each PM in order to compute the power model.

Meng et al., proposed a VM provisioning approach to consolidate multiple VM instances for the same PM in order to improve resource utilisation and thus reduce the energy consumed by under-utilised PMs [91]. A VM selection algorithm was developed to identify compatible VM instances for consolidation. Compatible VM instances are those with similar capacity demand, defined as their application performance requirement, and these are grouped into sets allocated to the minimum number of PMs. It can be argued that consolidating compatible VM instances to the same PMs will have a small negative effect on applications assigned to each VM instance and thus keep SLA requirements from being violated. The study found an improvement of 45% in resource utilisation.

The authors in [89] and [32] devised an algorithm to allocate VM instances to PMs at data centres with the goal of reducing power consumption in PMs without violating the SLA agreement between a Cloud provider and users. The researchers argued that assigning a group of VMs to as few PMs as possible will save power [116]. The energy-aware resource algorithm [86] has two stages: VM placement and VM optimisation. The VM placement technique aims to allocate VMs to PMs using a Modified Best Fit Decreasing (MBFD) algorithm. This is based on the Best Fit Decreasing (BFD) algorithm that uses no more than 11/9 * OPT + 1 bins (OPT is the optimal number of bins) [117].

The MBFD algorithm sorts VMs into descending order of CPU utilisation in order to choose power-efficient nodes first. The second stage is the optimisation step responsible for migrating VMs from PMs that are either over- or under-utilised. However, VM migration may cause unwanted overheads, so should be avoided unless doing so reduces either power consumption or performance, so the authors set lower and upper thresholds for utilisation. If the total utilisation of the CPU of a PMs falls below the lower threshold, this indicates that the host might consume more energy than it needs. Similarly, if the utilisation exceeds the upper threshold then the performance of the hosted VMs may deteriorate. In this case, some VMs should migrate to another node to reduce the level of utilisation. The authors concluded that the Minimisation of Migrations (MM) policy could save up to 66% of energy, with performance degradation of up to 5%. It was found that the MM policy minimised the number of VMs that have to migrate from a host in the event of utilisation above the upper threshold.

Graubner et al., proposed a VM consolidation mechanism based on a live migration technique with the aim of saving power in Cloud computing [115]. They developed a relocation algorithm that periodically scans available PMs to determine which PM to migrate VM instances from, and which PM to migrate them to. The approach was found to save up to 16% of power when implemented in the Eucalyptus platform, however the relocation process was unclear, with no further explanation of when it is triggered during run time [118].

The authors in [119] proposed GreenMap, a power-saving VM-based management framework under the constraint of multi-dimensional resource consumption in clusters and data centres. GreenMap dynamically allocates and reallocates VMs to a set of PMs within a cluster during runtime. There are four modules in the framework: clearing; locking; trade-off; and placement. The clearing module is responsible for excluding VMs inappropriate for dynamic placement, for instance those with unpredictable or rapid variation in demand. The locking module monitors SLA violations caused by the workload, in which event the module will switch to a redundant VM for execution. The trade-off module evaluates the potential of a new placement generated by the placement module in respect of performance and cost trade-off. The placement module performs a strategy for reallocating live VMs to another physical resource to save power, based on a configuration algorithm. The algorithm starts by randomly generating a new placement configuration. The placement module then delivers the configuration to the trade-off module. The experiment showed that it is possible to save up to 69% of power in a cluster, with some performance degradation, but it did not consider the overheads of the placement module.

The authors in [118] proposed an energy-saving mechanism developed and implemented for a private Cloud called Snooze, tested using a dynamic web workload. The authors argued that it differed from other power-aware VM mechanisms in two aspects, in that it was applied and tested in a realistic Cloud environment, and that it takes dynamic workload into consideration. A monitor unit was introduced periodically to check running PM; any under- or over-utilised nodes were reported to a general manager module to issue a migration command. There are four VM allocation policies: placement; overload relocation; underload relocation; and consolidation.

The placement policy allocates new VM instance requests to PMs using RoundRobin scheduling, which distributes the load to PMs in a balanced way. The overload policy scans PMs to check if a PM is overloaded with VM instances and, if so, searches for a PM that is only moderately loaded to accommodate these VM instances in all-or-nothing way (i.e. migrate all running VMs or none). The migration command is sent to the migration policy for straightforward execution. Similarly, the underload policy issues a migration command to migrate VMs from under-utilised PM in an all-or-nothing way. The mechanism managed to save up to 60% of power, the experiment concluded, but it was conducted in a homogenous infrastructure, that is, it assumed that all PMs have the same computing capacity. In addition, the all-or-nothing method may be a drawback as it leads to PMs being overloaded, which may cause performance degradation in instances of hosted VM.

3.2.3 Performance-Aware Approaches

Van et al., proposed a virtual resource manager focused on maintaining service levels while improving resources utilisation via a dynamic placement mechanism [10]. The manager has two levels: a local decision module and a global decision module. The first is concerned with applications, as the manager deals with complex N-tier levels in, for instance, online applications that require more than one VM instance to process. The global decision module has two stages: the VM placement stage, concerned with allocating a VM to a specific PM with the goal of improving resource utilisation; and the VM provisioning stage of scheduling applications to VMs (i.e. sending applications to be processed by VM instances).

The authors in [120] proposed a novel VM placement approach of two phases: candidacy and placement. The former elects a list of PMs eligible to accommodate VM instances, choosing the candidate PM on the basis of migration capability, network bandwidth connectivity and user deployment desire, which should be available beforehand. Available PMs have a four-level hierarchy representing an ordering system of PMs available to be candidates. The latter phase selects one of the candidate PMs from the first phase to host a VM instance on the basis of low-level constraints. The authors argue that the first phase can help to reduce the time spent choosing the most suitable PM. However, this work requires prior knowledge of user deployment of VM instances, which is not supported in CSPs. CSPs usually offer different classes of VM instances for end users to choose between. Asking further questions regarding user preferences is not economically viable.

The authors in [88] proposed a VM placement technique that employs the FF heuristic solution to maximise revenue for CSPs under performance constraints, expressed as an SLA violation metric measuring performance degradation of VM instances caused by using the FF mechanism to improve resource utilisation. The proposed system has two managers: the global manager decides which PM hosts a VM instance; and the local manager is concerned with scheduling VM instances within the hosted PM. The global manager employs a decision-making policy for each candidate PM's viability for hosting a VM instance in such a way as to improve resource utilisation.

Calcavecchia et al., proposed the Backward Speculative Placement as a novel VM placement technique [121]. The VM placement technique has two phases: continuous deployment and ongoing optimisation. The continuous deployment phase allocates a VM instance to the PM with the highest demand risk, a scoring function to measure the level of dissatisfaction with a PM at the final unit of time. It is, however, not clearly explained how this is awarded. The ongoing optimisation phase migrates VM instances hosted to a PM with high risk demand to another PM with a low score, as long it is able to accommodate the VM instances. The Backward Speculative Placement technique was able to decrease the execution time of submitted tasks.

3.2.4 Network-Aware Approaches

The authors in [99] proposed a VM placement and migration approach to minimise the effect of transfer time of data between VM instances and data storage. In Cloud computing, a CSP can provide VM instances to end users to process data while these data are stored in different locations, for example Amazon EC2 and Amazon S3. Therefore, the approach developed takes network I/O requirements into consideration when VM placement is applied. In addition, the VM migration policy is triggered when the time required to transfer data exceeds a certain threshold. Network instability is the main reason for this increase of time, and the threshold is stated in the SLA agreement. The study showed that the time taken to complete the task fell, on average, due to the placement of VM, depending on location.

A novel traffic-aware VM placement technique was developed by [122] with the goal of improving network scalability. The mechanism employs a two-tier approximate algorithm to place VM instances with PMs in such a way that significantly reduces the aggregate traffic in datacentres. The two-tier algorithm partitions VMs and PMs separately into clusters. The VMs and PMs are matched individually in each cluster. The partitioning step is achieved using a classical min-cut graph algorithm that assigns each VM pair with a high mutual traffic rate to the same VM cluster. Having VM instances with a high traffic rate in the same cluster of PMs means that traffic is exchanged only through that cluster, which can reduce the load upon switches at a data centre.

Purlieus [123] is a resource allocation tool developed to improve the performance of MapReduce jobs and to reduce network traffic by paying attention to the location of resources. MapReduce enables the analysis and processing of large amount of data in a quick and easy way [124]. Purlieus employs VM placement techniques that allocate VM instances to PMs according to their location. Purlieus was able to reduce the execution time of jobs by 50% for a variety of types of workload.

The authors in [125] studied the VM allocation problem from the network perspective [125]. They proposed a novel VM placement mechanism that considers network constraint, which is the variation in traffic demand time. Its goal is to minimise the load ratio across all network cuts by implementing a novel mechanism, the two-phase connected component-based recursive split, to choose the PM with which to place a VM instance. It exploits the recursive programming technique to formulate a ranking table of each VM instance that is connected. The PM with the least connected ranks of associated VMs is selected to host a new VM instance, but the proposed mechanism is for static VM placement only, thus it does not consider moving VM instances around during run time to reduce the cut load ratio.

The authors in [126] introduced S-CORE, a scalable VM migration mechanism to reallocate VM instances to PMs dynamically with the goal of minimising traffic within a datacentre. They showed that S-CORE can achieve cost reductions in communication of up to 80% with a limited amount of VM migration. S-CORE assigns a weight for each link in a datacentre, taking into consideration the amount of data traffic routed over these links. If the line weight exceeds a certain threshold, then some VM instances with high traffic load have to migrate to another PM using a different link. Such an approach avoids traffic congestion on core links at data centres to prevent any degradation in the performance of a Cloud system.

3.2.5 Failure-Aware Approaches

The aforementioned studies investigated various VM allocation mechanisms with the aim of minimising power consumption, improving performance or reducing the traffic load in Cloud systems. However, they all fell short of providing a mechanism tolerant of failure events in Clouds' PMs. Therefore, these VM allocation techniques are neither practical to employ nor to implement in a Desktop Cloud system. The following subsection reviews several studies that have tackled the issue of node failure.

A wide range of techniques and approaches has been developed to tackle node failure issues in Desktop Grid systems, because a node within a Desktop Grid system can voluntarily join or leave the system, increasing the probability of node failure, heightening the risk of losing results. For example, the authors in [127] developed a fault-tolerant technique in Desktop Grid systems that employs replication of applications to avoid losing them in failure events. Another approach was proposed by [128], based on the mechanism of application migration. This checks applications periodically during runtime, and in the event of node failures all associated application are restored and migrated to another node. However, this is not practical in this study because it is concerned with the applications level and violates the concept of the Cloud computing paradigm that isolates the infrastructure layer from the service layer to prevent CSPs from having control over services run by end users.

Machida et al., proposed a redundancy technique for server consolidation [129]. The focus was on complex online applications requiring several VM instance for each application, and the technique offers *k* fault tolerance with the minimum number of physical servers required for application redundancy [129]. It relies on replicating an application *a* times and running it for *k* number of VM instances. The number of VM instances is calculated on the basis of the requirements of application *a*, but requires full knowledge of and access to the applications and services that run on VM instances in order to replicate them. This, again, violates the concept of Cloud computing whereby CSPs are prevented from being able to access and control the applications of end users. Furthermore, the approach assumes that all physical servers have the same computing capacity, impractical in the era of Cloud computing where PMs are usually quite heterogeneous.

The authors in [130] proposed the BFTCloud, a fault-tolerant framework for Desktop Cloud systems that tackles the specific malicious behaviour of nodes known as Byzantine faults: machines that provide deliberately wrong results. The framework employs a replication technique with a primary node by 3 * *f*, where *f* is the number of faulty nodes at run time. The framework considers failure probability as the mean to choose primary nodes and their replicas in respect of QoS requirements. Byzantine faults are identified by comparing the results reported by a primary node with those of its replica; if the results are inconsistent then they will be sent to another node to process and compared to detect which machine is behaving suspiciously. However, the calculation of failure probability is not clearly given. In addition, although the framework was said to be for Desktop Cloud systems, it does not possess the essential feature of employing virtualisation to keep the service layer isolated from the physical layer; in fact, the technique is to replicate tasks by sending one to a primary node and its 3* *f* replicas of nodes. Another issue worth mentioning about the BFTCloud mechanism is the notion of *f*, which means that the number of faulty nodes should be known before run time. However, this technique is impractical since the number of node failures in such distributed systems is unpredictable and difficult to calculate [131].

The authors in [132] addressed the issue of node failure in hybrid Clouds, that is, private and public Clouds. The problem is formulated as follows: a private Cloud with limited resources (i.e. PMs) has a certain number of nodes with a high failure rate. The question is how to minimise the dependency of public Clouds to achieve better QoS, given that sending workload to a public Cloud costs more. The authors proposed a failure-aware VM provisioning for hybrid Clouds, a 'time-based brokering strategy', to handle failure of nodes in private Clouds by redirecting tasks required long term into a public Cloud. The decision to forward a task to a public Cloud is based on the duration of the request; if longer than the mean request duration of all tasks, then it will be forwarded. Although the proposed strategy considers that a public Cloud solves the issue of node failure in private Clouds, the issue is not answered unless the reliability of this public Cloud can be guaranteed.

The review of VM mechanisms in this section shows that the design of a fault-tolerant VM allocation mechanism remains an open research problem that needs to be tackled in Cloud environments with faults, such as in Desktop Cloud systems.

3.3 Summary

The VM allocation mechanism considers the placement of new VM instances to Cloud PMs and their migration around PMs to achieve optimisation objectives such as resource utilisation. This chapter explained the processes involved in designing a VM allocation mechanism. Several related areas of research were presented yet excluded, being beyond the scope of the VM allocation problem.

Table 3-1: Summary of Taxonomy of VM Allocation Mechanisms

Approach	**Focus**	**References**
Heuristic Solutions	Compare several heuristic solutions	[98],[107]
	Review Cloud management tools	[108],[109]
Power-Aware	Study the relationship between energy consumption, resource utilisation and performance in Traditional Clouds	[106]
	Improve resource utilisation	[91],[32],[115],[118]
	Reduce power consumption with acceptable effect upon performance	[96]
	Reduce power consumption with acceptable number of SLAs violations	[32],[119]
Performance-Aware	Improve resource utilisation with focus on performance metric	[10]
	Reduce execution time	[120],[121]
	Reduce the impact of VM degradation	[88]
Network-Aware	Reduce the transfer time between VMs	[99]
	Reduce traffic rate	[122],[123],[126]
	Minimise load ratio on networks	[125]
Failure-Aware	Fault-tolerant techniques for Desktop Grid systems	[127],[128]
	Replication techniques for Traditional Cloud systems	[129]
	Tackle the Byzantine fault issues	[130]
	Study node failures in hybrid Clouds	[132]

The chapter provided a literature review of works proposed to improve the outcome of Cloud systems. These can be classified into: studies employing a heuristic solution to tackle the VM allocation problem as a Bin Packing problem; studies proposing to reduce power consumption in Clouds' nodes; studies that try to improve the performance of Cloud systems; studies introduced to reduce the impact on network and bandwidth; and studies tackling the issue of node failure in Cloud systems. Table 3-1 provides a summary of reviewed works according to the taxonomy presented in this chapter.

However, this review of the literature reveals that none of these works are appropriate for implementation in a Desktop Cloud system that tolerates failure events in PMs, and the next chapter presents a novel VM allocation mechanism that is able to do so.

Chapter 4: A Fault-Tolerant VM Allocation Mechanism

The previous chapter reviewed the various techniques and mechanisms proposed to enhance the outcome of Cloud systems in terms of enhancing performance and reducing power consumption and network traffic, together with some fault-tolerant approaches. However, it revealed that none of these mechanisms can tackle the issue of node failure in Desktop Cloud systems. In Traditional Clouds, PMs are assumed to be of high reliability [4], making the possibility of node failure quite small. However, the case is different in Desktop Clouds because the nodes are expected to be highly volatile, as described in section 2.3.4.4. This chapter proposes a Fault-Tolerant (FT) VM allocation as a new mechanism to handle node failure during runtime.

This chapter starts by discussing the allocation problem from two perspectives: from research and from Desktop Cloud's experience of high failure rates in PMs. The former deals with the problem as an optimisation problem without sufficient attention to node failure, while the latter approach takes this into consideration. The proposed mechanism employs a replication technique to reduce the impact of node failure on throughput, yet this causes PMs to consume more power, so the FT mechanism incorporates a migration policy to reduce power consumption. The chapter concludes with throughput, power consumption and availability metrics that may be used to assess and evaluate the FT VM allocation mechanism.

4.1 VM Allocation Problem

In Cloud computing, VM instances are hosted and run in PMs, so if a PM fails, all allocated VM instances to it are destroyed. Consequently, PM failure means all the tasks running on the VM instance are lost, badly affecting the throughput outcome. In Desktop Clouds, the problem is crucial because the number of node failure events can be high, therefore the goal of VM allocation in this context is to design a mechanism that can improve the throughput of a Desktop Cloud system in the presence of node failure.

Multi-criteria optimisation problem is a problem that is evaluated by two or more conflicting criteria. In the context of VM allocation problem in Desktop Cloud system, the problem is formulated as multi-criteria through two factors: throughput and power consumption. Throughput is improved using fault-tolerant mechanism using a replication technique which increases the power consumed by nodes. Therefore, the mechanism is extended to reduce power consumption with an accepted penalty on throughput outcome. One way to overcome the issue of node failure is to replicate VM instances, so that if a VM instance is destroyed there is a replica available to continue processing tasks. However, there are several obstacles when designing such a mechanism. Consolidation of new requested VMs to improve resource utilisation is considered a challenge in Traditional Clouds [104]. Another challenge in Traditional Clouds is the migration of running VM instances with an aim to improve performance or resource utilisation [108]. Summarised, the challenges in designing a VM allocation mechanism for Desktop Cloud systems based on [82] and [92] are:

Challenge 1. Replication of VM instances with the aim of reducing the impact of node failure on throughput.

Challenge 2. Selection of a PM to host a replicated VM.

Challenge 3. Response to a failing PM.

Challenge 4. Implementation of a migration policy of VM instances during runtime to reduce the impact of the replication technique.

Therefore, the FT mechanism is designed to take into consideration the aforementioned challenges. The first two tackle the proposed mechanism. This creates a VM replica for each requested VM and consolidates it to a PM already hosting a VM. The third challenge is dealt with by the FT mechanism, locating a VM replica of the destroyed VM in order to make it primary and, in turn, to create a replica. The last challenge requires the FT mechanism to employ a migration policy that aims to reduce power consumption by improving resource utilisation. Therefore, the FT mechanism incorporates a novel policy termed the Utilisation-Based Migration Policy (UBMP), a dynamic live migration policy to migrate VM instances to PMs to improve resource utilisation and thus reduce the power they consume in a Desktop Cloud system.

4.2 Overview of Cloud Management Platform

An overview is given in this section to explain how the proposed mechanism can be employed in a Desktop Cloud system. In Cloud computing, end users can submit their tasks to a Cloud system to be processed in a VM or list of VM instances. A Cloud scheduler distributes assigning tasks to VM instances, managed by an allocator. In IaaS Cloud systems, the Cloud allocator receives requests from users who may stipulate the number of desired VM instances along with the specification of each VM instance. The allocator employs a VM allocation mechanism that places, migrates and replicates VM instances.

Figure 4-1 depicts an overview of a management platform to which a scheduler sends tasks submitted by users to VM instances, managed by an allocator. The allocator keeps monitoring the running VM instances while they process tasks until released by users. In the Cloud system, there is a pool of PMs ready to accommodate VM instances. Several open-source Cloud management platforms exist, such as OpenStack [133] and Eucalyptus [12], that are able to be integrated with the proposed VM allocation mechanism to manage a Desktop Cloud system.

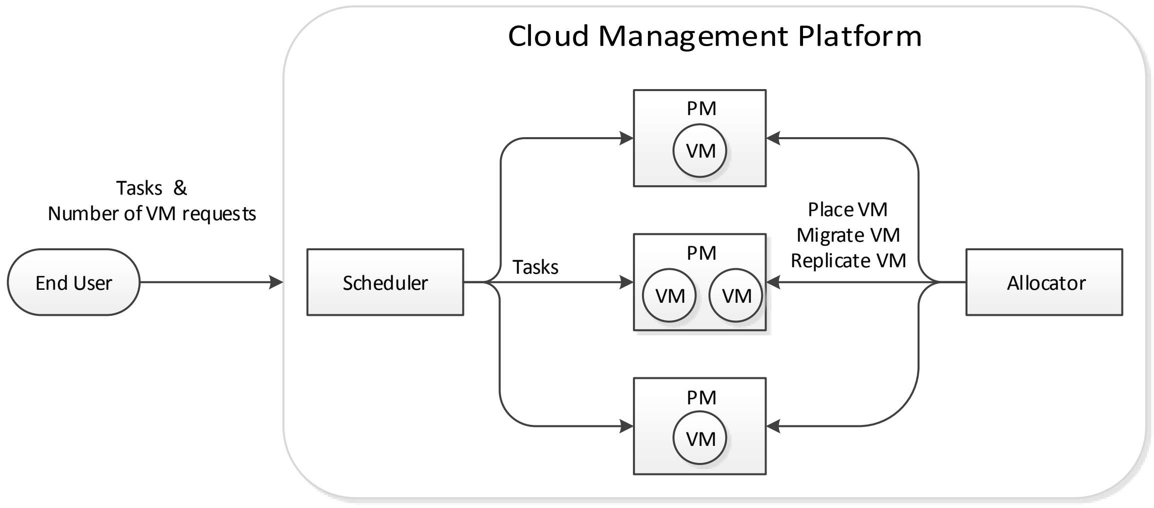

Figure 4-1: System Overview

The proposed mechanism aims to improve throughput in the presence of node failure by replicating VM instances when requested by Cloud users, incorporating a migration policy that aims to improve resource utilisation.

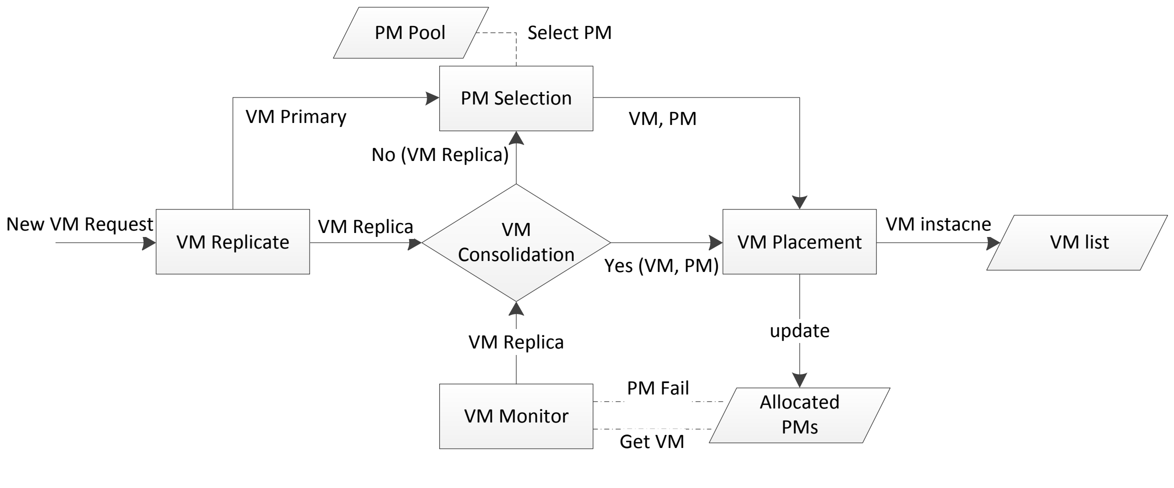

Figure 4-2: FT Mechanism

4.3 FT Mechanism

The FT allocation model is depicted in Figure 4-2. The mechanism is a failure-tolerant technique that works by replicating the requested VMs. VM instances are created and hosted on a PM and remain working there until released by the user who requested them. If a PM is reported as failing, the mechanism responds by locating a replica in order to make it primary and creating a new VM replica. The mechanism starts by receiving from a user a new VM request that specifies its required computing capabilities (CPU and RAM). The request is sent to the *VM Replica* module, which creates two VM instances: a primary VM that is sent to the *PM Selection* module to select a PM host it, and a VM is replica that is sent to the *VM Consolidation* module to try to allocate the replicated VM onto a PM already hosting another VM(s). The following explains each module of the mechanism further.

4.3.1 VM Replication

The *VM Replication* module is responsible for replicating the requested VM in order to improve the resilience of the mechanism in the event of failure. When a new request arrives, the module will replicate the request to create two VM versions: one a primary VM and the other a VM replica. The replication technique is based on the idea that more than one version of the VM will run simultaneously. There are two constraints for this technique: the first is that the VM has to be hosted by a PM that does not host the primary VM. This is to ensure that there is at least one version of the VM running in the event of failure. The second constraint is that it is necessary to try to consolidate the replicated VM instance with another VM instance to improve resource utilisation, given the replication technique’s disadvantage of increasing the power consumed by PMs, so if possible the replica has to be allocated to a PM already hosting a VM(s).

This method will improve resource utilisation and minimise the effect on the availability of PMs. When the *VM Replication* module receives a VM request, it sends it to the *PM Selection* to choose a PM to host the primary version of the new VM. The replica is sent to the *VM Consolidation* module to consolidate it to a PM already hosting a VM. The *VM Replication* module ensures that a VM and its replica will not be assigned to the same PM by assigning VMs the same VM identity. This is to let the *VM Consolidation* module recognise them and thus avoid assigning a replica to the primary VM. VM replicas are updated frequently by the *VM Monitor* to keep them synchronised with their primary VMs.

```
set pmList = list of available PMs

vm = getVmRequest()
selectedPM = pmList.get(0)

for i = 1 to pmList.number
     tmpPM = pmList.get(i)
     if  ((utility(tmpPM) < utility(selectedPM)) &&
          (suitableToHost(tmpPM, vm) == true) )
               then selectedPM = tmpPM
     end if
end for

return selectedPM
```

Figure 4-3: PM Selection Policy

4.3.2 PM Selection

The *PM Selection* module is responsible for the selection of a PM from a pool of available PMs to host a VM request. The LF heuristic solution is employed to choose a PM with the least utilisation. The selected PM has to be able to accommodate this VM, and the policy is shown in Figure 4-3. The step is required to minimise the number of affected primary VM instances in the event of PM failure. The *PM Selection* module may receive another request to select a PM for a VM replica, because the *VM Consolidation* module could not consolidate it to a utilised PM. Once the PM is picked up, it and its VM will be sent to the *VM Placement* in order to finish the allocation process.

```
set workingPmList = list of working PMs

replicaVM = getVmRequest()
consolidatedPM = null

for i = 0 to workingPmList.getSize()
     tmpPM = workingPmList.get(i)
     if ((utility(tmpPM) > utility(consolidatedPM)) &&
          (suitableToHost(tmpPM, replicaVM) == true) )
               then consolidatedPM = tmpPM
     end if
end for

return consolidatedPM
```

Figure 4-4: VM Consolidation Policy

4.3.3 VM Consolidation

The *VM Consolidation* module is responsible for the allocation of a VM replica to a PM. The VM replica request is sent from the *VM Replicate*. The *VM Consolidation* module is a decision-making policy that employs the BF heuristic solution to select a PM with the highest utilisation level to host the VM replica, and the policy is described in Figure 4-4. The motivation for this step is to improve utilisation and minimise the number of running PMs. If a no VM instance is hosted to a PM then the PM is put onto a power save mode to reduce power consumption. This approach is demonstrated to be effective in saving power in Traditional Clouds, according to [86].

However, it is not always possible to find a PM that is already hosting a VM (or list of VMs) to host the replica, since there is a possibility that all candidate PMs are unable to accommodate the replica as they do not have sufficient computing capability. The *VM Consolidation* module obtains a list of all PMs that host VMs, then a decision is made on whether to allocate the replica to one of the listed PMs. If this is affirmative (i.e. a PM is found), then the *VM Placement* module will be informed and will allocate the replica to the identified PM. If the answer is negative, no PM can host it, so a new request will be sent to the *PM Selection* module in order to select a new one.

4.3.4 VM Placement

The *VM Placement* module simply allocates the received VM to the received PM. The *Allocated PMs* list contains all working PMs; that is, all PMs hosting at least one VM instance. The module updates the *Allocated PMs* list with this PM if it does not already contain the PM. For example, if the sent VM and PM come from the *VM Consolidation* module, then the received PM has already been added to this. The *VM Monitor* module updates the list if a PM becomes free (i.e. no VM is hosted to it), and the PM is removed. The *VM list* contains all VM instances currently running. The created VM instance is ready to process tasks submitted by end users in a Desktop Cloud system. If a VM is released by its user, then the *VM Monitor* module removes it from the *VM list*.

4.3.5 VM Monitor

The *VM Monitor* module scans and monitors VM instances while they are being used to process tasks and it has several roles. The first is periodically to checkpoint [134] primary VM instances in order to keep replicated VMs well synchronised with their primary instances, using an asynchronous replication technique proposed by [135]. This keeps replication overheads to the minimum. When a VM is released by its user, it will be destroyed and removed from the *VM list*. If a working PM becomes idle, thus not hosting a VM, then the module will remove it from the *Allocated PMs* list.

Another role is periodically to check the status of each running PM. If a PM fails, it means all VMs associated with this PM are also destroyed. The VM instances can be either VM primaries or replicas, so the *VM Monitor* will obtain the information from the *VM list* in order to provide replicas. If the destroyed VM is a replica, then the module will create a new VM replica. If the destroyed VM is a primary, then the module will make its replica a primary and create a new replica. In both cases, the *VM Monitor* will send a replication order for each destroyed VM to the *VM consolidation* for replication. This step is important to avoid reaching *complete failure* status, which means failure of a primary VM and its replicas at the same time. However, there is still a risk of *complete failure* status if the PM that hosts a primary VM fails at the same time as the PM that hosts the replica.

4.4 Utilisation-Based Migration Policy

The previous section introduced the FT mechanism that focuses on improving throughput by replicating VM instances. However, the mechanism lacks the ability to migrate VM instances dynamically around PMs with the aim of improving resource utilisation and thus reducing power consumption. This section enhances the FT mechanism by implementing a Utilisation-Based Migration Policy (UBMP), a dynamic policy that migrates VM instances to improve utilisation with only a minimal effect on the throughput outcome of Desktop Cloud systems. The mechanism is depicted in Figure 4-5. The mechanism remains the same as described in the previous section apart from the *VM Monitor* and the *VM Migration*, which are explained in the forthcoming subsections.

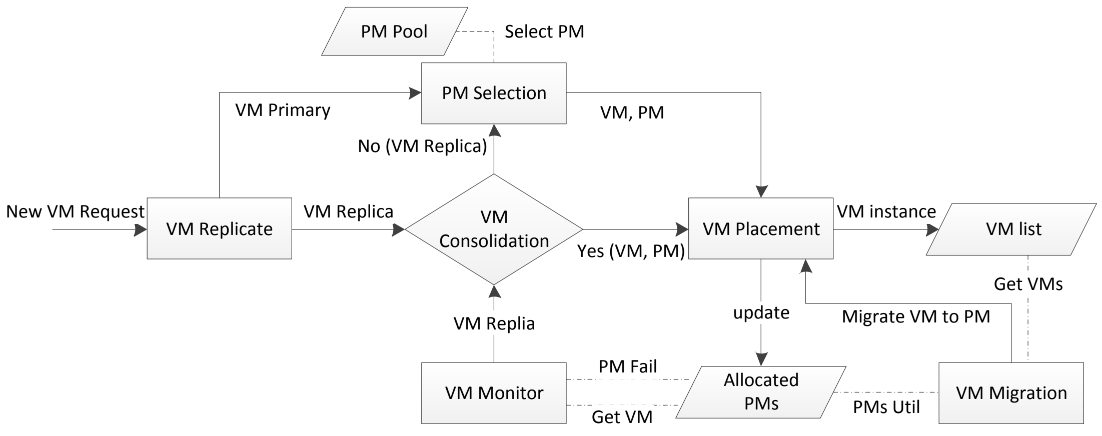

Figure 4-5: Dyncamic FT Mechanism with UBMP

4.4.1 VM Monitor

The role of the *VM Monitor* module is extended in this FT mechanism to be able to issue migrate orders to the *VM Migration* module to move VM instances from under-utilised PMs to other working PMs. If a working PM fails, the module responds by replicating VM instances that are allocated to that PM. The replicated VM instances are sent to the *VM Consolidation* module to have PMs selected for them. In addition, the *VM Monitor* periodically scans all working PMs to let the *VM Migration* module move VM instances from one PM to another. It keeps the *VM Migration* module updated with the lists of working PMs and running VM instances.

4.4.2 VM Migration

The *VM Migration* communicates with the *VM monitor* to obtain a list of the PMs hosting VMs to migrate VM instances. The UBMP is depicted in Figure 4-6. If the utilisation level of a PM is below a threshold, termed *util*, all VMs are set to be migrated to another candidate PMs. These candidate PMs are those whose utilisation levels are above *util*. These are sorted in descending order according to their utilisation level and, if able to accommodate a VM instance, the highest is selected from all PMs with a utilisation level below *util*. If not capable (because the CPU or available RAM is inadequate for a VM instance or a particular PM does not host a version of the VM instance) then the next PM from the list is checked. This process is repeated for all PMs with utilisation levels below *util* and stored in an ‘under-utilised PMs list’. The UBMP stops this mechanism in two cases: the first is if no PMs from the candidate list is able to host a VM allocated to a PM from the under-utilised PMs list, and the second is if there is no PM on the under-utilised PMs list because all VM instances have been migrated to other PMs in the candidate list.

```
underutilisedPMsList = getUnderutilisedPM(uti)

for i = 0 to underutilisedPMsList.getSize()
     underutilisedPM = underutilisedPMsList.get(i)

     //get list of VMs allocated to this PM
     migratingVMsList =
underutilisedPM.getAllocatedVMslist()

     //list of PMs that can accommodate
     candidatePMsList = getCanidatePMsList(uti)

     // sort PMs according to utilisation
     sort(candidatePMsList)

     candidatePM = getPMtoMigrate(candidatePMsList,
migratingVMsList)

     if (candidatePM != null) then
          migrate(candidatePM)

     //no more candidate PMs
     else then
          end for

     end if
end for
```

Figure 4-6: UBMP

The utilisation threshold plays a key role in the trade-off between resource utilisation versus throughput, because consolidating many VM instances to the same PM increases the risk of complete failure status, where the primary and its replicated VM instances are destroyed at the same time when their hosted PMs fail simultaneously. Therefore, it can be a challenge to find the utilisation threshold that provides both resource utilisation and minimal reduction of throughput. In this context the utilisation threshold is based on a ‘try and check’ basis by setting various threshold levels and comparing the results, as sections 5.3.1 and 6.3 explain.

A live migration approach is employed in this migration policy. Alternatively, there are non-live migration approaches and techniques involving a process of suspension of a VM instance, then migration to the destination PM and finally resumption [136]. However, this is unsuitable because non-live migration approaches can take longer than live, increasing the risk of a PM failing during migration, so they are inappropriate for fault-prone systems such as Desktop Clouds. Nevertheless, the migration of VM instances incurs the penalty of performance degradation during runtime. According to [137], during migration this can rise to 10% of the performance of the VM. The FT mechanism employs the Xen [138] hypervisor deployed in Desktop Clouds’ PMs because it enables VM live migration. The maximum acceptable performance degradation when a VM instance is being migrated is regarded as 10% for this study.

4.5 Evaluation Metrics

The efficiency of Cloud computing is determined by a set of evaluation metrics. Employing efficient metrics for Cloud computing is vital in order to optimise the outcome of Cloud systems. It has been shown that there is no systemic analysis of evaluation metrics for Cloud Computing [139]. The diversity of architectures of Cloud providers requires that good performance metrics are platform-independent [140]. However, the literature shows that several studies have assessed the service provided by the Cloud from the prospective of consumers. Most of the literature (such as by [141], [142] and [143]) focuses on the cost-performance of services in order to adopt a better decision-making policy to help customers choose a service provider according to their requirements. For example, some customers can tolerate some performance degradation in exchange for a low cost service.

From the perspective of the service provider, there is a need for an evaluative approach to assess the behaviour of the infrastructure used. The evaluation approach can be employed to improve the quality of service [109] or reduce maintenance costs [118]. Evaluation approaches of virtualisation technology cannot be employed in Cloud computing, because they are restricted to their hypervisor and do not aim to measure the performance of IaaS Clouds [141]. Various works have been proposed in the literature to evaluate the outcome of VM mechanisms, such as: power consumption, as in [106]; network traffic, as in [99]; and SLA violations, as in [10]. The evaluation metrics are proposed to investigate and evaluate the outcomes of a Desktop Cloud system, namely throughput, power consumption and availability.

4.5.1 Throughput

Most studies in the literature focus on the performance notion, including attributes such as response and average turnover time, such as [144] and [142]. Researchers fail to focus on node failure because they assume that PMs in Cloud systems are highly reliable [145] and, since most VM allocation mechanisms and techniques rely on PMs in Cloud systems being thus, throughput is not studied despite being an important measure of the outcome of a Cloud system in the presence of node failure. The throughput metric calculates the ratio of successfully completed tasks submitted by end users for execution in a Cloud system to the total number of submitted tasks [146], calculated as follows:

$$throughput = 100 * \frac{number\ of\ successful\ tasks}{total\ number\ of\ tasks} \tag{1}$$

4.5.2 Power Consumption

The power consumption metric considers the amount of energy consumed by each PM in the infrastructure layer of a Cloud system [119]. The authors in [86] use power consumption as one of the metrics of the outcomes of their energy-aware mechanism for Cloud computing. Energy efficiency can be defined as the number of instructions, in billions, executed per Watt-hour [147]. The Standard and Performance Evaluation Corporation (SPEC) community released a SPECpower metric of power consumption [148]. SPECpower is a Java application that generates a set of transactions completed per second, calculating the energy consumed from the total number of operations in Watt-hours. Energy consumption is considered a metric for evaluating the FT mechanism in Desktop Clouds, given as follows:

$$power\ consumption = \sum_{i=0}^{n} power(PM_i) \tag{2}$$

where *power* is a function using the SPECpower metric to calculate the current power consumption of a PM according to its utilisation level. *Power consumption* is, in this thesis, measured in kilowatt-hours (kWh). Cloud computing promises to improve resource utilisation through resource multiplexing (i.e. assigning more than one VM to the same PM). It has been shown that there is a linear relationship between utilisation and power consumption [149], therefore measuring power consumption reflects better resource utilisation on the part of the FT mechanism. Better resource utilisation means fewer PMs to host VM instances, and PMs with no VM instances (i.e. with 0% of resource utilisation) are switched to a power saving mode in order to preserve energy.

4.5.3 Availability

Availability means how much computing power of PMs in a Cloud system is available to accommodate new VM requests. Evaluating the availability of PMs can reflect the impact of using a replication technique. In addition, failure of nodes can affect the availability of Desktop Cloud systems. A question in this context is whether the employed VM allocation mechanism can help in improving node availability. The availability is calculated as the ratio of computing power available for each PM against the maximum computing availability of all PMs. The availability is given as follows:

$$availability = \frac{\sum_{i=0}^{n} availability(PM_i)}{total\ availability}$$

4.6 Summary

A novel fault-tolerant VM allocation mechanism designed to tolerate the node failure prevalent in Desktop Cloud systems was presented in this chapter. The FT mechanism is intended to ensure that the impact of node failure on the throughput of a Desktop Cloud system is kept to the minimum, and is enhanced by incorporating a migration policy based on utilisation level that reduces power consumption in Desktop Cloud systems with only minimal impact on throughput. The proposed mechanism can be evaluated by three metrics: throughput; power consumption; and availability.

The chapter started by explaining the problem of allocation mechanisms. The problem has been discussed before in the literature but is here extended to improve throughput. The throughput outcome of a Desktop Cloud can be adversely affected by node failure events. Several challenges of designing a fault-tolerant mechanism were discussed and an overview given of a platform to manage a Desktop Cloud system. The FT mechanism was designed to improve throughput by implementing a replication technique for VM instances, however this can lead to higher power consumption by PMs, so it was extended to minimise this negative impact. The Utilisation-Based Migration Policy (UBMP) was designed for the FT mechanism to reduce its power consumption.

The introduction of three metrics concluded the chapter: throughput; power consumption; and availability. Throughput is the number of tasks successfully completed during runtime. The availability metric is employed to capture the effect of the replication technique employed by the FT mechanism. The metrics will be used in this study to assess the FT mechanism through simulation experiments.

The next chapter presents the methodology of this research into the implementation and evaluation of the proposed mechanism.

Chapter 5: Research Methodology

Previous chapters reviewed and discussed the research gaps and issues in Cloud computing. Section 2.3 showed that the infrastructure level of Desktop Cloud systems comprises nodes that are volatile and prone to failure without prior notification, making it challenging to ensure that throughput in a Desktop Cloud system remains at an acceptable level. Chapter 4 proposed a new VM allocation mechanism that employs a replication technique running VM instances to reduce the negative impact of node failures on throughput in Desktop Cloud systems.

This chapter presents the methodology of this research to evaluate and test the proposed fault-tolerant VM mechanism. It starts by explaining the potential of using simulation as a mean of evaluating VM allocation mechanisms using three metrics: throughput; power consumption; and availability. CloudSim is a widely used simulation tool developed to simulate Cloud systems with the aim of demonstrating several Cloud issues and evaluating the proposed solutions. However, node failure events are not simulated in CloudSim. DesktopCloudSim was developed in this study to overcome this issue by allowing for the augmenting of failure in a Cloud’s nodes during runtime.

Furthermore, real traces of node failures of Desktop Grid systems were collected from the Failure Trace Archive (FTA), an online repository providing an archive of several distributed systems. The selected Desktop Grid systems are NotreDame and SETI@home systems. The traces of such Desktop Grid systems can be read by DesktopCloudSim in order to simulate the behaviour of nodes in a Desktop Cloud system. An analysis of the FTA traces is proposed and discussed next to demonstrate that node failure in such systems can be quite high, thus it is an issue that needs attention. The design of experiments conducted in this research explains how they are undertaken, including: the specification of a simulated Desktop Cloud system; the workload of tasks submitted to get the simulate system working; the number and details of requested VM instances; and how the evaluation metrics (throughput, power consumption and availability) are calculated during the simulation.

The final section presents a baseline experiment conducted to answer the following question:

- *What is the difference between Desktop Clouds and Traditional Clouds in terms of throughput, power consumption and availability?*

The experiment is carried out by implementing three VM allocation mechanisms: the FCFS; Greedy; and RoundRobin mechanisms, as explained in section 3.2.1. Each was run for four different Cloud scenarios: the NotreDame Desktop Cloud system; NotreDame Traditional Cloud system; SETI@home Desktop Cloud system; and SETI@home Traditional Cloud. The results are analysed and discussed in the discussion subsection.

5.1 Simulation

Simulation means the simplified imitation of a set of processes comprising a particular system over a period of time in order to understand or improve that system [150]. Researchers find it difficult to predict the behaviour of systems due to complexity, leading to the use of simulation [151]. These can serve several purposes such as education, training, entertainment and experimentation [150].

Simulation may be preferable to conducting experimentation for several reasons. First, studying the effect of modifications on real systems is likely to be more expensive than conducting experiments in a simulated environment. However, simulations can still be costly and may require extra computing power [151]. Second, simulation can save a great deal of time, especially when investigating the effects of different modification on a specific aspect in a real system. Added to this, simulation can produce results that would take a long time, maybe years, to be obtained in a real system. Third, researchers have more control over simulated environments, allowing them to adjust conditions for a better understanding of specific aspects of the proposed change. Finally, in many cases the targeted system does not yet exist, making simulation the practical alternative.

Many leading organisations as well as many researchers around the world use simulation of Cloud Computing to study multiple issues and proposed solutions to optimise Clouds. For example, HP Labs in USA uses Cloud simulation tools to investigate resource provisioning and energy efficient techniques in data centres [14]. The next subsection describes CloudSim as the desired simulation tool for conducting experiments this research.

5.1.1 CloudSim

CloudSim is a Java-based discrete event simulation toolkit designed to simulate Traditional Clouds [152]. A discrete system is a system whose state variables change over time at discrete points, termed ‘events’ [153]. The tool was developed by a leading research group in Grid and Cloud computing, the CLOUDS Laboratory at the University of Melbourne in

Australia.[5] The simulation tool is based on both the GridSim [154] and SimJava [155] simulation tools.

GridSim enables modelling and simulation of Grid computing to allow the study of possible ways and techniques to improve the effectiveness and performance of Grids. It is expensive and time consuming to build a ready-to-use Grid infrastructure [154], moreover, researchers with access to such infrastructure are limited to a certain number of resources and domains, thus the tool is of benefit. GridSim is based on the SimJava environment, while SimJava is a graphical user interface for simulating and modelling complex systems based on the discrete event simulation kernel [155], enabling the construction of discrete event simulation models for fairly generic systems.

Table 5-1: CloudSim vs. Grid Simulation Tools

Simulation Tool	**Multi-Layering**	**Virtualisation**	**Popularity**	**Stability**
SimGrid	Not supported	Not supported	Widely used	Stable
GroudSim	supported	supported	Limited use	In progress
CloudSim	supported	supported	Widely used	Stable

CloudSim tool was selected as the simulation toolkit to conduct experiments in this study for various reasons. First, CloudSim is claimed to be more effective in simulating Clouds than SimGrid [156] and GroudSim [157], as illustrated in Table 5-1. The reason is that CloudSim allows segregation of multi-layer service abstraction (i.e. IaaS, PaaS and SaaS services). This is an important feature of CloudSim that most Grid simulations do not support. Researchers can study each abstraction layer individually without affecting others. Second, virtualisation is much employed in the Cloud paradigm, unlike Grid, therefore the majority of Grid simulations do not take virtualisation of resources into consideration, making them a less attractive choice [152]. Third, CloudSim is implemented in a highly modular way that makes it extensible for further modification. Finally, CloudSim is reviewed and updated frequently to detect any bugs. The current and stable version of CloudSim is version 3.0.3, an indication that the tool is quite stable. It is used and tested by various researchers from different research groups and domains, including industry.

[5]http://www.cloudbus.org/

Table 5-2: Comparison of Cloud Simulation Tools

Simulation Tool	Release Date	Programming Language	Open Source	Energy Efficiency
CloudSim	2009	Java	√	Supported
MDCSim	2009	Java	×	Supported
GreenCloud	2010	C++	√	Supported
iCanCloud	2012	C++	√	Not supported

Table 5-2 lists the most common Cloud simulation tools along with their release date and the programming language for implementation: MDCSim [158]; GreenCloud [159]; and iCanCloud [160]. The table also shows whether the mentioned simulation tool is open source or not. Energy efficiency in the table means whether or not the simulation tool enables the design and investigation of power-saving solutions.

MDCSim is a commercial, discrete-event simulation tool developed at Pennsylvania State University to simulate multi-tier data centres and complex services in Cloud computing. It has been designed with three-level architecture, including a user-level layer, a kernel layer and communication layer for modelling the different aspects of a Cloud system. MDCSim can analyse and study a cluster-based data centre with in-depth implementation of each individual tier. The tool can help in modelling specific hardware characteristics of different components of data centres such as servers, communication links and switches. It enables researchers to estimate the throughput, response times and power consumption. However, as the simulation tool is a commercial product, it is unsuitable to run experiments, such as this research.

GreenCloud is another cloud simulation framework, implemented in C++ and focused on the area of power consumption and its measurement. The tool was developed on top of Ns2, a packet-level network simulation tool [161]. Having the tool implemented in C++ makes it feasible to simulate a large number of machines (100,000 or more), while Java is assumed to be able to handle only 2GB memory on 32 bit machines. However, CloudSim was able to simulate and instantiate 100,000 machines in less than 5 minutes with only 75 MB of RAM, according to [15]. Although GreenCloud can support a relatively large number of servers, each may have only a single core. In addition, the tool pays no attention to virtualisation, storage and resource management. Such limitations make it an undesirable choice to run experiments for this research.

iCanCloud is a C++ based open source Cloud simulation tool based on SIMCAN [162], a tool to simulate large and complex systems. It was designed to simulate mainly IaaS Cloud systems, such as instance-based clouds like EC2 Amazon Cloud. iCanCloud offers the ability to predict the trade-off between performance and cost of applications for specific hardware to advise users about the costs involved. The tool has a GUI feature and can be adapted to different kinds of IaaS Cloud scenarios. However, iCanCloud does not enable researchers to study and investigate energy efficiency solutions, making it unsuitable for this research.

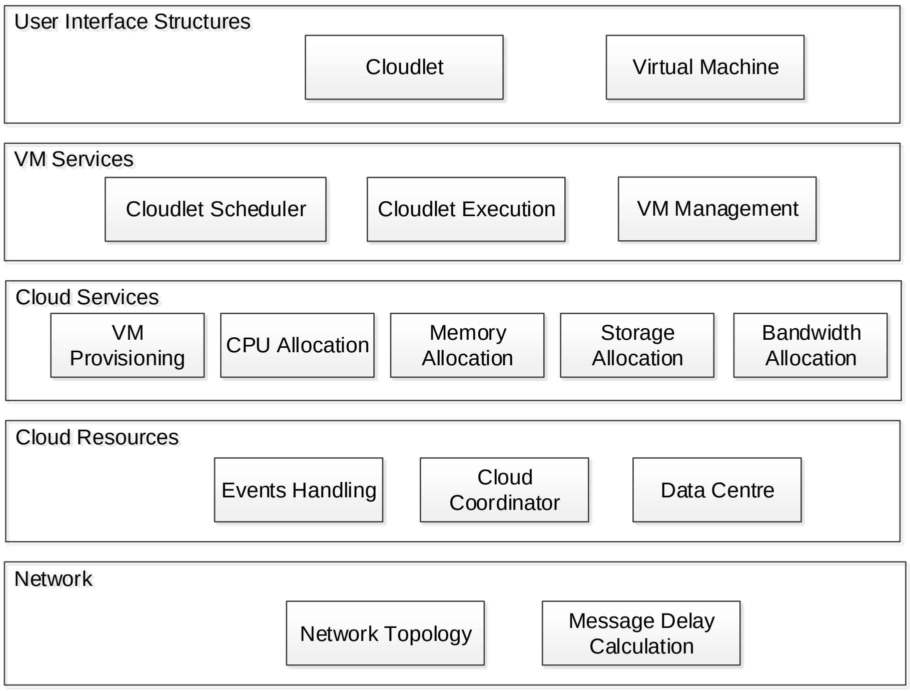

Figure 5-1: CloudSim Architecture

Figure 5-1 shows the architecture [152] of CloudSim, simulating the environment of a Cloud system. The architecture contains several layers ranging from network level to service layers. *User Interface Structures* layer contains a *cloudlet* that represents Cloud-based application services processed by the *Virtual Machine*. This layer simulates the role of a Cloud broker in serving and providing Cloud services to end users, as it is a SaaS Cloud. In this layer, end users can submit their tasks to be processed and executed in VMs. Tasks are simulated by CloudSim in the form of cloudlets, each of which is assigned a computing processing value, termed ‘cloudlet length’, expressed in millions of instructions (MI). The PaaS Cloud is simulated in the *VM Services* layer. The layer is in charge of execution requests by users, in the *Cloudlet Execution* component, as well as managing VMs for them by *VM Management*. Each VM instance has computing power in millions of instructions per second (MIPS) and RAM capabilities. Each VM instance processes a range of cloudlets, and the time required to execute and finish each depends on the *Cloudlet Scheduler*.

The *Cloud Services* layer is the stage for modelling the VM allocation mechanism. Researchers can extend this layer in order to investigate their VM allocation mechanisms in data centres. It simulates the management of providing the VM instances in the *VM Provisioning* component. Each VM instance is allocated to a PM; a *CPU Allocation* computing power in MIPS; *Memory Allocation* in gigabytes; and *Storage Allocation* size in gigabytes.

The *Cloud Resources* layer contains *Events Handling*, which handles events passing between simulation entities such as hosts, data centres and Cloud brokers. Each *Data Centre* component can contain a number of host coordinated by the *Cloud Coordinator*. The *Cloud Resources* layer can be extended in order to simulate Desktop Clouds by changing the behaviour of physical nodes. The *Network Layer* simulates the network level in Cloud systems, which allows simulating *Network Topology* such as links and switches. Communications and messages between network elements are simulated in *Message Delay Calculation*.

Table 5-3: CloudSim Entities

System Entity	**Role**
Cloudlet	Simulation of services and applications
VM	Simulation of virtual machines instances
PM	Simulation of physical machines
Broker	Simulation of Cloud brokers
Data Centre	Simulation of data centres

CloudSim is comprised of a range of entities (listed in Table 5-3: CloudSim Entities) that represent a component or module within a Cloud system. Each entity can contact other entities by sending events to the targeted entity. Each event is triggered at a given time to be executed during run time, and serves a particular task.

Table 5-4: Host Features in CloudSim

Specification	**Measure**
CPU	MIPS
Number of Cores	Integer number
RAM	Gigabyte
Storage	Gigabyte
Bandwidth	Megabyte

A Cloud's PM is simulated in CloudSim as a host entity, which has several features as listed in Table 5-4. A VM instance is simulated in CloudSim as a VM entity. In CloudSim, a Cloud broker can request and set the number of VM instances during run time. The features, mentioned in Table 5-5, of a VM instance can be set by Cloud brokers.

Table 5-5: VM Features in CloudSim

Specification	**Unit**
CPU	GB
RAM	GB
Storage	MB
Bandwidth	GB

CloudSim tool offers several features for researchers. The features are: (1) simulation and modelling of large-scale Cloud systems on a single computer machine; (2) a platform to simulate and control separately components in Cloud environment such as data centres, brokers, services, scheduling policies and VM allocation mechanisms; and (3) support modelling of network connections among and between simulated system entities.

Although CloudSim is considered the most mature Cloud simulation tool [15], it falls short by failing to provide several important features. It does not simulate the performance variations of simulated VMs when they process tasks [163]. Second, service failures such as those in tasks during running time and complex overhead of complicated tasks are not simulated [164]. Furthermore, it lacks the ability to simulate dynamic interaction of nodes in the infrastructure level; it only allows static configuration of nodes that remain without change during run time. Lastly, node failures are not included in the CloudSim tool.

A simulation is suitable for this research because there is no actual Desktop Cloud on which to run experiments. In addition, the simulation enables control of the configuration of the model to study each evaluation metric. In this research, CloudSim is used to simulate the resource management model. CloudSim allows altering the capabilities of each host machines located in the *data centre* entity in the simulation tool. This feature is highly useful in experimentation, as it is needed to set the infrastructure (i.e. physical hosts) to be of an unreliable nature. Extending CloudSim to enable simulation of node failures instead of building a new simulation of Desktop Clouds can serve the research community by enabling other researchers to use the extended tool for their research. In addition, it further helps to make CloudSim tool the desired simulation tool by encouraging researchers to use or extend it for experiments.

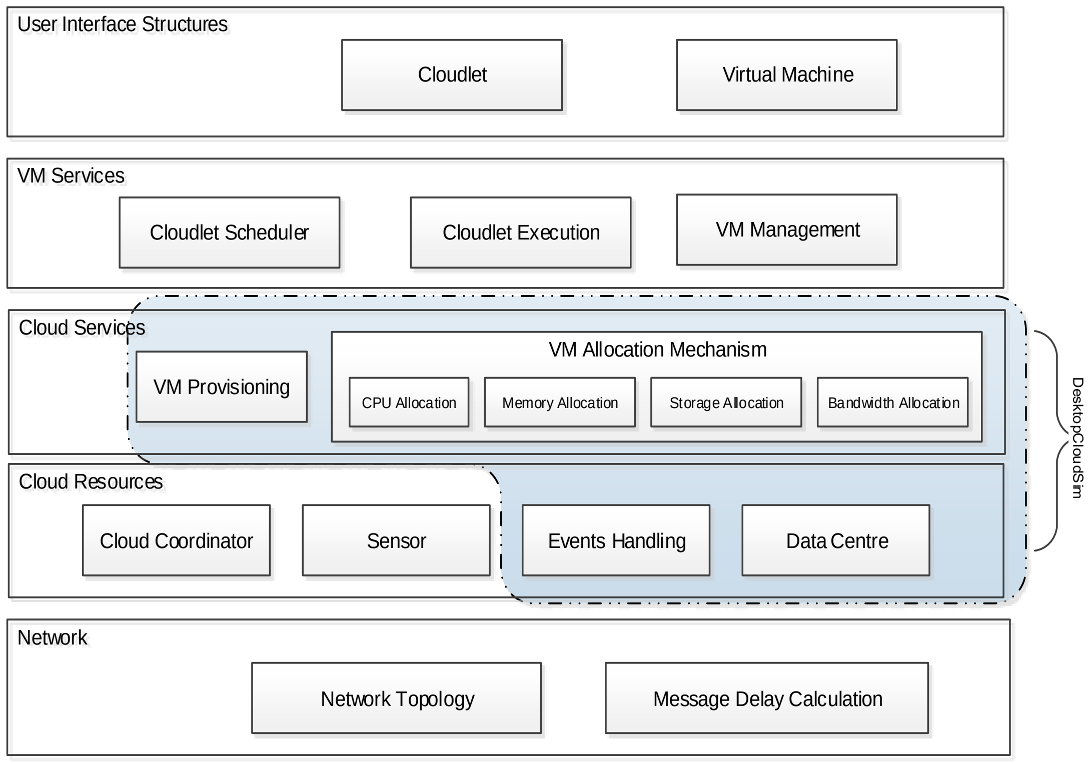

Figure 5-2: DesktopCloudSim Abstract

5.1.2 CloudSim Extensions

There are several extensions of CloudSim that have been developed to overcome the limitations of CloudSim tool. The extensions are NetworkCloudSim [165], WorkflowSim [164], DynamicCloudSim [163], FederatedCloudSim [166] and InterCloud [167]. NetworkCloudSim is an extension simulation tool based on CloudSim to enable the simulation of communication and messaging aspects in Cloud computing. The focus of the tool is on the network flow model for data centres and network topologies, bandwidth sharing and the network latencies involved. It also enables the simulation of complex applications such as scientific and web applications that require interconnections between them during run time. Such features can allow further accurate evaluation of scheduling and resource provisioning mechanisms in order to optimise the performance of Cloud infrastructure.

WorkflowSim is a new simulation extension that has been published recently as an extension for CloudSim tool. The tool was developed to overcome the shortage of CloudSim in simulating scientific workflow. The authors of WorkflowSim added a new management layer to deal with the overhead complex scientific computational tasks, arguing that CloudSim fails in simulating the overheads of such tasks such as queue delay, data transfer delay, clustering delay and postscripts. This issue may affect the credibility of simulation results. They also point out the importance of failure-tolerant mechanisms in developing task scheduling techniques. WorkflowSim focuses on two types of failures: tasks failure and job failure. A task contains a number of jobs, so failure in a task causes a series of jobs to fail. However, our work differs from WorkflowSim in the failure event and its impact. The focus of this research is on the infrastructure level, containing nodes hosting VMs, whereas its authors were interested in the service level, that is, tasks and applications. It can be argued that service providers should consider developing failure-tolerant mechanisms to overcome such events in the infrastructure level.

DynamicCloudSim is another extension for CloudSim tool. Its authors were motivated by the fact that CloudSim lacks the ability to simulate instability and dynamic performance changes in VMs during runtime. This can have a negative impact on the outcome of computational intensive tasks, which are quite sensitive to the behaviour of VMs. The tool can be used to evaluate scientific workflow schedulers, taking into consideration variance in VM performance. In addition, the execution time of a given task is influenced by the I/O-bound such as reading or writing data. Its authors extended instability to include task failure. Performance variation of running VMs is an open research challenge, but beyond the scope of this study.

FederatedCloudSim [166] is an extension tool in the CloudSim toolkit to enable the simulation of federated Clouds using difference federation scenarios, while respecting SLAs. According to Goiri et al., Cloud Federation is the idea of bringing many CSPs together in order to avoid the case of over-demand for Cloud services by letting a CSP rent out CSPs to other computing facilities [168]. FederatedCloudSim enables researchers to simulate and study various ways to standardise interfaces and communications between CSPs in a federated Cloud. Such a tool can help to study optimisation solutions for exchanging Cloud services between CSPs without violation of SLAs. InterCloud is another simulation tool that has been developed to simulate Cloud federation, based on the CloudSim tool. However, InterCloud falls short of providing sufficient simulation capabilities of SLAs, compared to FederatedCloudSim.

5.1.3 DesktopCloudSim

DesktopCloudSim is an extension tool developed by this research to simulate failure events in the infrastructure level based on CloudSim simulation tool. Simulation is necessary to investigate issues and evaluate solutions in Desktop Clouds because there is no real Desktop

Cloud systems available to run experiments. In addition, simulation enables control of the configuration of the model to study each evaluation metric. In this research, CloudSim is extended to simulate the resource management model. CloudSim allows the alteration of the capabilities of each host machine located in the *data centre* entity of the simulation tool. This feature is useful in experimentation, as it is needed to set the infrastructure (i.e. physical hosts) as unreliable. This can be achieved by extending the *Cloud Resources* layer in the simulation tool. Figure 5-2 depicts the layered architecture of CloudSim combined with an abstract of the DesktopCloudSim extension.

DesktopCloudSim extends CloudSim tool with version 3.0.3, the most recent version of CloudSim that is stable until 2015. Thanks to the high flexibility of CloudSim, DesktopCloudSim does not modify the original code of CloudSim tool but provides a new package, implemented in Java, of new classes that often extend those of CloudSim.

DesktopCloudSim extended CloudSim by adding the following modules: “Data Centre”, “Events Handling”, and “VM Provisioning”. The “Data Centre” module was modified by adding “Host” as being a new “SimEntity” that can be modified dynamically during run time. This makes the simulation of node failing possible in DesktopCloudSim. “Events Handling” module was also extended by adding new events to simulate adding or removing PMs during run times. In addition, some VM events were added as well such as “VM replicate” in order to enable simulation of VM replication technique. The module of “VM Provisioning” was modified by implementing a novel allocation policy called “DesktopAllocationPolicy” that implements the FT mechanism proposed by this research. In addition, “VM monitor” module was extended by adding new monitoring policies that work with new allocation policy to let VM instances migrate from a PM to another. In addition, the monitor module enables destroying VM instances during run time as a result of node failure.

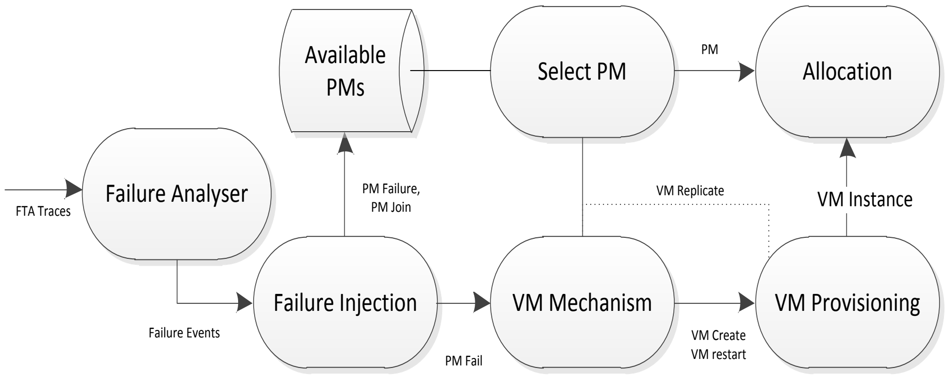

Figure 5-3: DesktopCloudSim Model

Figure 5-3 illustrates the components of DesktopCloudSim that read FTA trace files, as explained later in this chapter. The trace files contain the failure events of PMs. The *Failure Analyser* component analyses the files of failures to send failure events to *Failure Injection* component. The *Failure Injection* component receives failure events from the *Failure Analyser* and inject failures into associated PMs during run time by sending events to *Available PMs* component. The *Available PMs* contains a list of PMs that are ready to be used, so if a PM fails then it is removed or, if a PM joins, it is added. The *Failure Injection* component informs the *VM Mechanism* unit if a PM fails, to let it restart the failed VMs on another live node or nodes. The *VM Provisioning* component provisions VMs instances to be allocated to PMs selected by *Select PM*. The *VM Mechanism* controls which PM hosts a VM instance. The *VM Mechanism* creates restart VM instances. In addition, the *VM Mechanism* can replicate a running VM instance, if required.

5.2 Analysis Method

The results of experiments were analysed using SPSS V21, because it is well-accepted and widely used as a statistical analysis tool by many researchers in different disciplines [169]. The Kolmogorov-Smirnov (K-S) test of normality was used to determine whether the studied data follow a normal distribution or not. The test was used because it is suitable for a medium number of samples, as was the sample size of 180 samples, as explained later in section 5.3.2. If the critical value (p-value) obtained from the normality test was greater than 0.05, it indicated that the data were normally distributed, otherwise the data did not follow a normal distribution.

The results of each experiment were tested using adequate statistical tests. First of all, all tests were repeated measures because the same input data, which is node failure events, were used for every experiment. If the results of an experiment were normally distributed, then the parametric test repeated t-test was used to compare the results of the two VM mechanisms. If the results were not normally distributed, then the non-parametric Wilcoxon test was applied to compare the results of the two mechanisms.

In case there were comparisons between more than two VM mechanisms, there were two steps to test the results. If the results followed a normal distribution, then the first was to conduct the parametric repeated one-way ANOVA test to show whether there was a statistically significantly difference. If there was, then the second step was to run a number of post-hoc tests, repeated t-tests, to compare each pair of VM mechanisms. The number of post-hoc tests was calculated as the number of compared VM mechanisms -1. The critical value p-value was corrected using the Bonferroni correction method [169] which is 0.05/number of post-hoc tests.

If the results did not follow a normal distribution, then the non-parametric Friedman's test was applied to compare the results of more than two VM mechanisms to see if there was a statistically significantly difference. If the test showed a difference, then a number of post-hoc tests were used to compare each pair of VM mechanisms, with the Bonferroni correction method was applied as explained before. However, in this case the post-hoc tests were conducted using the non-parametric Wilcoxon test because the results were not normally distributed.

5.3 Experimental Design

This section explains the design of all experiments conducted in this research because they share the same design. However, the next chapter presents the setting of each experiment in separate sections, apart from these common aspects. The next subsection presents the experiments conducted to evaluate the work of this research, along with the associated hypotheses. Next, the actual traces of node failures used in the simulation are presented. These data were studied to illustrate the percentage of node failures on an hourly basis. The specification of Desktop Cloud’s nodes, tasks of workload and the VM instances are then presented because they remained the same in all conducted experiments.

5.3.1 Experiments and Research Hypotheses

There were three experiments conducted in this research:

- *Experiment I: The Impact of Node Failure*: the first experiment was conducted to demonstrate that throughput, power consumption and availability metrics can evaluate the impact of node failures on the outcome of Desktop Clouds. It investigated which VM allocation mechanism of the FCFS, Greedy, RoundRobin and Random mechanisms yielded the best results for each evaluation metric. The experiment was to test the following hypothesis:

> *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds*

- *Experiment II: Evaluation of the FT Mechanism*: the second experiment evaluated the proposed VM allocation mechanism in improving the throughput of a Desktop Cloud system compared to other VM mechanisms tested in the first experiment. The experiment compared the FT mechanisms in terms of yielding the best results in each evaluation metric from the previous experiment. The experiment was to test the following hypothesis:

  > *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud System*

- *Experiment III: Utilisation-Based Migration Policy*: the third and last experiment of this research investigated a way to reduce power consumption of nodes when the FT mechanism is employed. The FT mechanism implemented a replication technique to ensure that, if a VM instance is destroyed because its hosted PM fails, there is another copy of this VM instance working on another PM. However, this replication technique can lead nodes to consume more energy. The experiment tested the following hypothesis:

  > *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with an accepted decrease in the throughput outcome*

Another experiment was conducted, as presented and discussed at the end of this chapter. It was a baseline experiment conducted to show that the DesktopCloudSim is capable of simulating node failures by comparing the results of three VM mechanism (FCFS, Greedy and RoundRobin) when employed in a Desktop Cloud compared to when employed in a Traditional Cloud. Such an experiment can illustrate the difference in outcome between Desktop Clouds vs. Traditional Clouds in terms of throughput, power consumption and availability metrics. It is worth mentioning that all of the aforementioned experiments simulate IaaS Cloud systems which was explained in section 2.2.2.

5.3.2 Failure Trace Archive Data Set

The FTA[6] is a public repository containing traces of several distributed and parallel systems [170]. The archive includes various systems including Grid computing, Desktop Grid, peer-to-peer (P2P) and High-Performance Computing (HPC). The archive contains timestamp events that are recorded regularly for each PM in the targeted system. Each event has a state element that refers to the state of the associated PM. For example, an event state can be unavailable, which means the PM is down at the time of the event. The unavailable state is considered a failure event throughout the experiment. The failure of a PM in an FTA does not necessarily mean that this node is down. For example, a PM in a Desktop Grid system can be become unavailable because its owner decides to leave the system at that time. Table 5-6 shows the four state definitions for PMs. A PM is considered a failure in the experiment if it is unavailable, used by its owner or it is over-utilised.

Table 5-6: PM State

State Code	FTA Definition	Experiment Definition
0	Unavailable	Failure
1	Available	Not failing
2	User present	Failure
3	CPU threshold exceeded	Failure

The FTA provides several traces for Desktop Grid systems at the University of Notre Dame, SETI@home, UC Berkeley CAD, San Diego Supercomputer Centre and University of Paris South. However, this work is limited to two systems: Notre Dame and SETI@home FTAs, because they contain sufficient failure traces for PMs. There are other data sets of Cloud systems that can be used to simulate Cloud systems, for example trace files provided by Google [171]. However, these data sets are not appropriate for this research because they cannot be used to simulate Desktop Cloud systems since the reliability of nodes provided by these sets is quite high.

[6]http://fta.scem.uws.edu.au/

Table 5-7: Number of PMs per Month

Month	Number Of PMs
1	432
2	479
3	503
4	473
5	522
6	601

The Notre Dame FTA was collected from the University of Notre Dame. The trace represents an archive of a pool of heterogeneous resources that ran for six months during 2007 [26]. Each month is provided separately, representing the behaviour of PMs located in the University of Notre Dame. The number of PMs varied from one month to another. Table 5-7: Number of PMs per Month lists the number of PMs of each month in the NotreDame FTA. The second trace archive is SETI@home FTA. The FTA has a large pool of resource (more than 200 thousand nodes) that were run for a year in 2008/09 [172]. The nodes in SETI@home are highly heterogeneous because most of these computing nodes are denoted by the public over the Internet. The archive of SETI@home contains traces of more than 100,000 nodes. However, a few of them have sufficient data to simulate a Cloud system for six months. Therefore, 875 nodes has been selected from SETI@home FTA which are those with trace files with sufficient failure events to simulate SETI@home Desktop Cloud for six months among other traces of SETI@home nodes.

Although the FTA archive provides traces of the behaviour of PMs, it needs some analysis to calculate the failure events. Several researchers have studied the failure events in the FTA archive, such as [26], [131], [173] and [174]. The literature shows that the focus is on the availability time behaviour of PMs; availability, in this context, means the time a machine remains able to serve. Studying the behaviour of nodes can discover a statistical model of availability in Desktop Grids [131] that can help predict the availability of machines to improve PM selection mechanisms, as mentioned in [173]. However, these works are not suitable for this research for two reasons. The first is that the focus is on the number of machines that fail, rather than the availability time. Second, the literature considers a machine as failed when it becomes unavailable, however in this simulation a machine is considered failed when it becomes unavailable or when its owner uses the machine, as mentioned above. Therefore, an analysis of node failure events of NotreDame and SETI@home systems is presented.

Failure means the percentage of nodes that fail in a given hour (h) divided by the total number of nodes in an FTA, as follows:

$$failure\ (h) = \frac{number\ of\ failed\ PMs\ at\ \ h}{total\ number\ of\ PMs} * 100$$

Study of the failure events is needed in this study to allow their simulation in a Desktop Cloud, since both Desktop Grids and Desktop Clouds can harness similar nodes. In addition, analysis of nodes failures can yield some benefits in optimising failure tolerance techniques. For example, failure events can help choose the appropriate number of replicas created for each primary VM instance, using the technique.

Table 5-8: Descriptive Results for Node Failures of NotreDame and SETI@home Systems

	NotreDame	**SETI@home**
N	4320	4320
Mean	6.26 %	13.67 %
Median	4.66 %	12.47 %
Std Dev	5.61 %	5.84 %
Minimum	0 %	3.43 %
Maximum	84.86 %	76.77 %

Table 5-8 shows a summary of results of analysis failure traces for NotreDame and SETI@home traces for six months. The table shows a summary of 4320 hours (6 months * 30 days * 24 = 4320 hours) including mean, median, standard deviation (Std Dev), minimum and maximum failure ratio in each hour. The Kolmogorov-Smirnov normality test was conducted on NotreDame and SETI@home results to show if they were normality distributed. The test showed that both were significantly different from normal distribution, $p < .05$. Two methods were employed to find the best distribution to fit the distribution of failure analysis results. The first method was a visual method including P-P plots and Probability Density Function (PDF). The second method was by conducting the goodness of fit test, which includes the Kolmogorov-Smirnova, the Anderson-Darling and Chi-Squared tests.

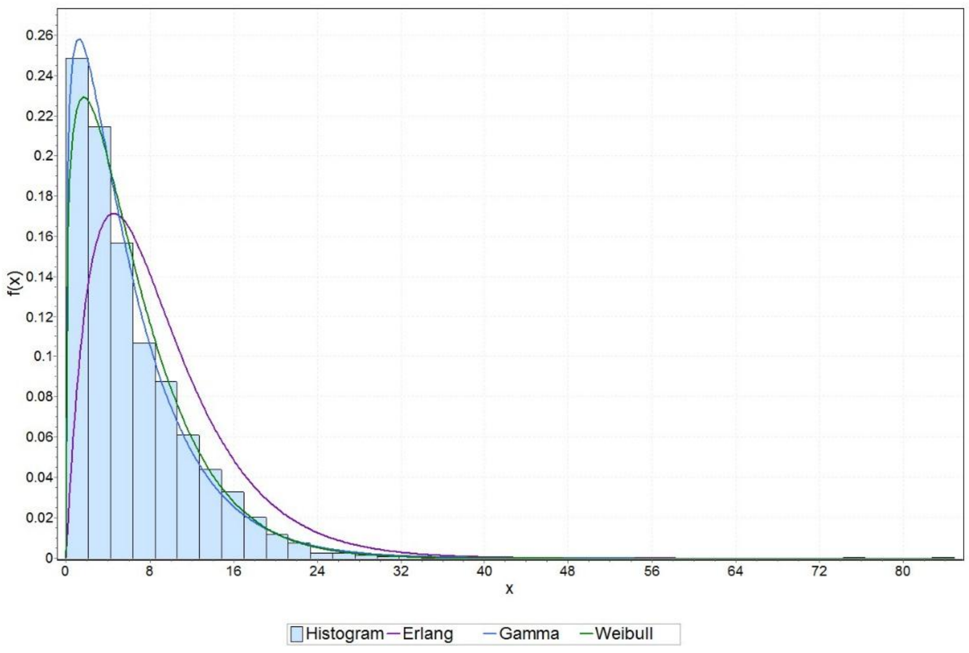

Figure 5-4: PDF Function for NotreDame Failure Analysis

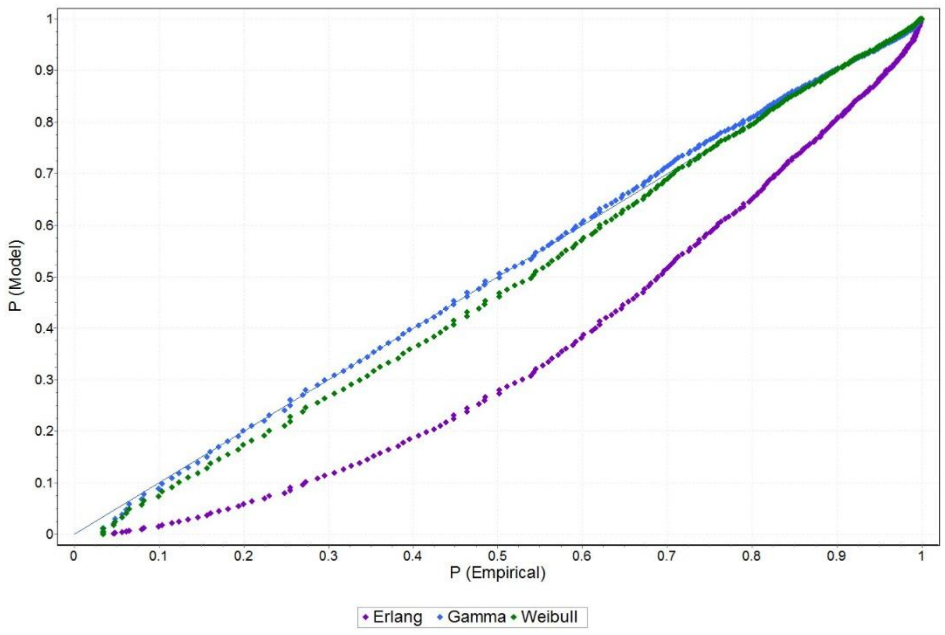

Figure 5-5: P-P Plot for Distribution Fitting for NotreDame Failure

NotreDame failure analysis results were tested against the Gamma, Weibull and Erlang distributions. Both Figure 5-4 and Figure 5-5 show visually that the Gamma distribution fits better than the others. It is worth mentioning that x in Figure 5-4 means the percentage of nodes failing among working nodes in hour h, while f(x) means the cumulative distribution of the percentage x [151] . In other words, f(x) represents the probability of nodes failing at variable x in hour h. For example, the probability of about 8% of NotreDame nodes to fail in a given hour is about 0.1 as it is showed in Figure 5-4.

Table 5-9 confirms this finding, because all three goodness of fit tests show that the Gamma distribution fits the data best, shape = 1.2434 and scale = 5.0343.

Table 5-9: Goodness of Fit Tests for NotreDame FTA Analysis Results

Distribution	Kolmogorov-Smirnov		Anderson-Darling		Chi-Squared	
	Statistic	Rank	Statistic	Rank	Statistic	Rank
Gamma	0.03403	1	259.45	1	15.885	1
Weibull	0.04693	2	272.14	2	68.331	2
Erlang	0.23122	3	1017.3	3	1861.9	3

Similar to the NotreDame failure results, SETI@home failure analysis results were tested against the Gamma, Weibull and Chi-Squared distributions. Both Figure 5-6 and Figure 5-7 show visually that the Gamma distribution fits better than other distribution. Table 5-10 confirms this finding because all three goodness of fit tests show that Gamma distribution fits the data best, shape = 3.3951 and scale = 3.0377. These findings are consistent with previous studies such as [175] and [174] which confirm that Gamma distribution is the best fit of failures in Desktop Grid systems.

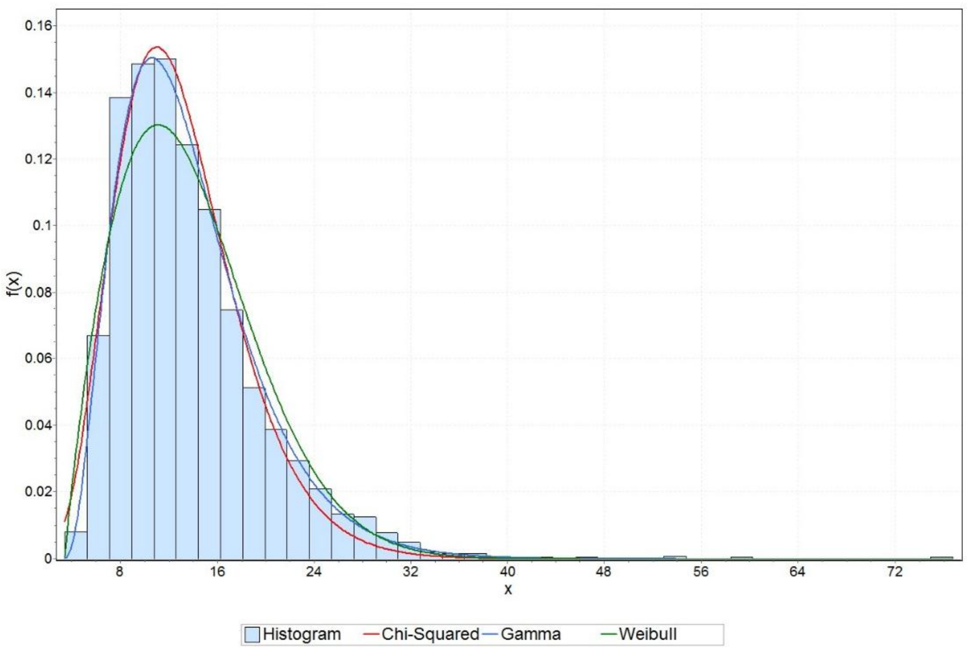

Figure 5-6: PDF function for SETI@home Failure Analysis

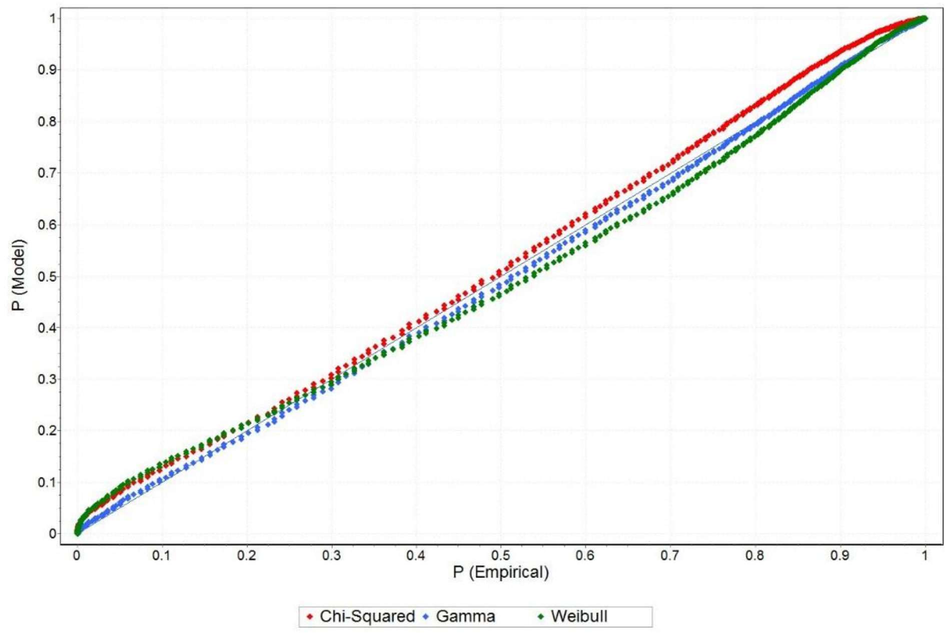

Figure 5-7: P-P Plot for Distribution Fitting for SETI@home Failure

Table 5-10: Goodness of Fit Tests for SETI@home FTA Analysis Results

Distribution	**Kolmogorov-Smirnov**		**Anderson-Darling**		**Chi-Squared**	
	Statistic	**Rank**	**Statistic**	**Rank**	**Statistic**	**Rank**
Gamma	0.02339	1	3.0501	1	38.142	1
Chi-Squared	0.03887	2	26.204	3	150.33	2
Weibull	0.04577	3	22.334	2	185.62	3

Another way is to look at the average failure percentage is as hourly failure over a period of 24 hours. This means calculating the average failure percentage of NotreDame machines during h=1, h=2... h=24 over a period of six months. The same applies to SETI@home. This analysis can give an idea of the behaviour of machines in the NotreDame and SETI@home Desktop Grids. Figure 5-8 and Figure 5-9 show the percentage failure range every hour of the day for the NotreDame and SETI@home failure traces. For example the minimum failure percentage of machines in NotreDame Desktop Grid in Hour 1 was 0%, while the maximum was 23%, as depicted in Figure 5-8. The NotreDame failure analysis shows that the range was 0% to 31% per hour, on average. Hours 15 and 14 recorded the highest failure at 84% and 75%, respectively. The failure range for SETI@home was more consistent, 6% to 30% of machines failing per hour. Hour 1 was exceptional, between 10% and 77%.

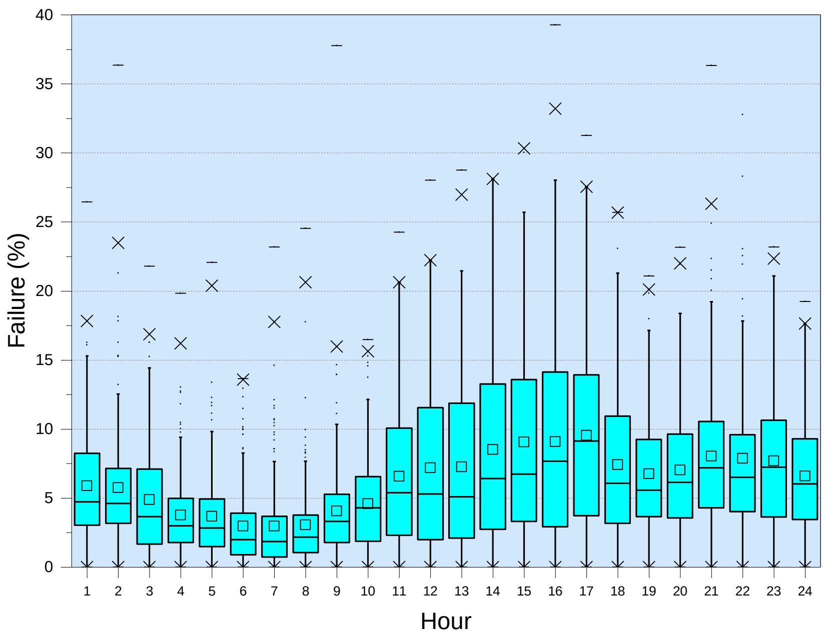

Figure 5-8: Failure Range for NotreDame FTA

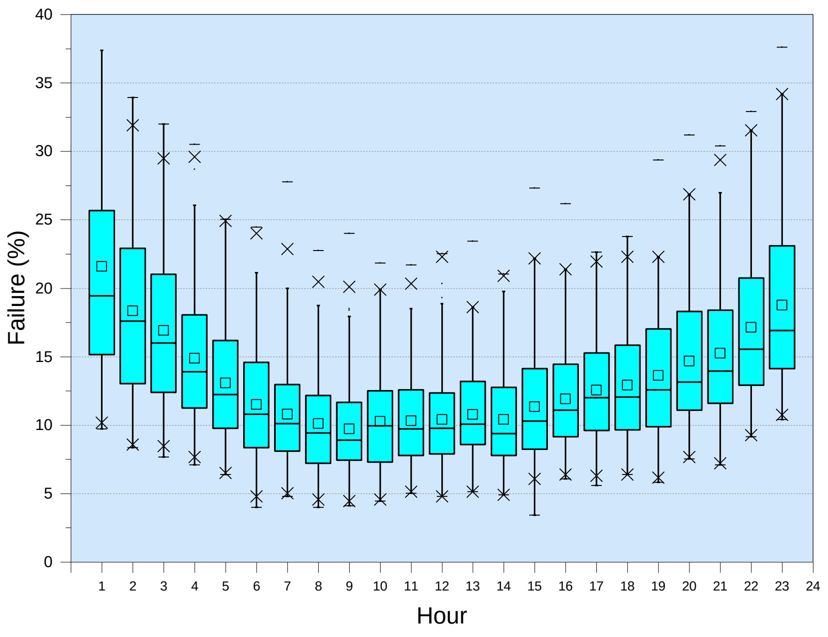

Figure 5-9: Failure Range for SETI@home FTA

Figure 5-10 depicts the average percentage of machines failing during a particular hour of the 24 hour period over six months for both NotreDame and SETI@home. For NotreDame, the lowest mean machines failure was in Hour 6, at 3%, while the highest was in Hour 17, at just above 9%. For SETI@home, the lowest figure was 9.5%, in Hour 9, and the highest was 21.5% in Hour 1. Overall, on an hourly basis, it seems that machines in the SETI@home Desktop Grid tended to fail more than the NotreDame machines. This is confirmed by the average percentage of machines that failed per hour: 13% of SETI@home and 6% of NotreDame. Connectivity issues are seen to be the main reason why failure events of nodes in SETI@home run at more than double the average in the NotreDame system; nodes in the latter are connected mainly by the local network, which makes the probability of a node going down less than its counterpart in SETI@home system, which is mainly connected over the Internet.

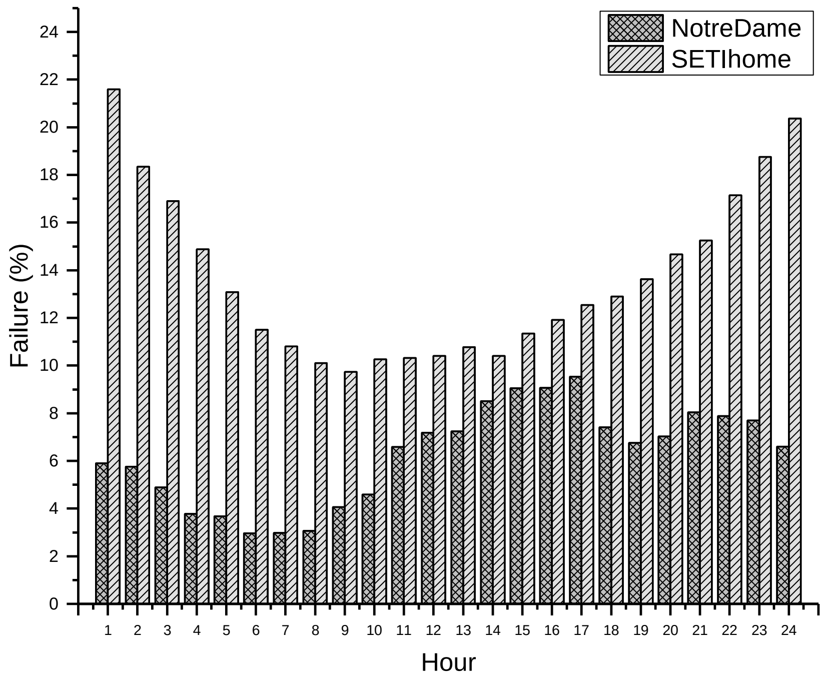

Figure 5-10: Average Failure Percentage for NotreDame and SETI@home FTAs

In short, this section presented a failure study of two Desktop Grid systems: the NotreDame system, considered an example of private system; and the SETI@home system, considered a public system. This study demonstrated that about 6.26% of nodes in a private Desktop Cloud system are expected to fail per hour of working time and about 13.67% of nodes per hour in a public Desktop Cloud system. Therefore, it is recommended to consider node failure as an issue when developing a VM allocation mechanism. Also, this study can help to predict the number of node failures, which can help in devising a policy to migrate VM instances from PMs that are expected to fail.

5.3.3 Hardware Specification of PMs

Although the FTA archive provides failure events for PMs, it does not offer sufficient detail regarding their hardware specifications, only reporting the number of PMs along with event traces of failure. It is quite important for this research to have different capabilities of machines for two reasons. The first is to evaluate various VM mechanisms accurately. For example, Greedy mechanism will behave just like FCFS when allocating VM instances if there is no different capabilities of each PM. The second reason is to evaluate the power consumption of nodes when the power varies according to utilisation level of PMs. If there is no diversity of specifications of PMs, this may mislead the judgement of power penalty of using replication technique. The capabilities are assigned randomly because the FTA archive does not provide hardware specifications of PMs so the random assignment can simulate the specification of PMs of NotreDame and SETI@home systems. Therefore, a list of several common computer machines have been collected online for use as PMs of the simulated Desktop Cloud systems. Table 5-11 gives the machine brand name and hardware specification, namely the CPU in MIPS, number of cores, RAM, hard and disk. The selected machines are quite heterogeneous in order to provide a similar infrastructure to that of a Desktop Cloud system, which is expected to be diverse. The specifications of PMs remained unchanged in all experiments conducted. Each run of an experiment and the hardware specification of the PMs were read from a text file.

Table 5-11: Hardware Specifications of PMs

Machine Brand Name	**Machine Specification**			
	CPU (MIPS)	**Core Number**	**RAM (GB)**	**Hard Disk (GB)**
Colfax International CX2266-N2	2400	4	16	400
Dell PowerEdge 2950 III	2833	8	16	73
Fujitsu Siemens Computers PRIMERGY TX150 S5	2666	2	4	160
HP Proliant DL160 G5	3000	8	16	80
Intel Platform SE7520AF2 Server Board	3600	2	4	36
SuperMicro 6025B-TR+	3000	4	8	36

In addition, each machine was assigned a benchmark of power consumption according to its utilisation level, from 10%, 20% up to 100%, namely the SPECpower benchmark, technical name SPECpower_ssj2008, developed by the SPEC [148]. Power consumption varies from machine to machine on the basis of this benchmark and each in the table has a listed consumption for certain utilisation levels, retrievable online from the SPEC website.

5.3.4 Task Workload

DesktopCloudSim needs tasks to be submitted in order to be processed and executed in a group of VM instances provisioned by a Desktop Cloud system. Therefore, to simulate these tasks in the experiments, tasks were collected online from PlanetLab and analysed by CoMon tool [176]. PlanetLab tasks were submitted for 24-hour run time on 03/03/2011, and the sum of tasks was submitted in the form of cloudlets at every hour of the 24-hour run time to analyse their failure percentage on an hourly basis. There were 1052 of cloudlets submitted every hour for a period of 24 hours (i.e. during a single run of each experiment). Cloudlet length was randomly assigned to the values reported in Table 5-12.

Table 5-12: Cloudlet Length

Cloudlet Length (MI)
$180 * 10^4$
$252* 10^4$
$360* 10^4$
$432* 10^4$
$504* 10^4$
$576* 10^4$
$648* 10^4$

However, the number of tasks and details remained the same during the runtimes of all experiments, because the focus of this research was to study the impact and behaviour of different VM mechanisms. Therefore, setting the task workload the same without changing was helpful in obtaining more accurate results of VM mechanisms and thus acquiring a better understanding. During every 24-hour run time, DesktopCloudSim read the task data as a list of cloudlets from a directory, containing them to ensure they remained the same.

5.3.5 VM instances

Open Virtualisation Format (OVF) is an open standard for packaging, distributing and describing VM hardware characteristics [177]. The OVF is described for being open, secure, portable, efficient and extensible set of descriptive files. Some characteristics that are described by OVF files can include CPU, memory, network cards, bandwidth and hard disks. However, this research focusses on CPU, RAM memory and disk when considering allocation. Although there are other features to be considered. For example, VM bandwidth is an important feature to be considered. Resource bandwidth cannot be simulated in this research because there are no actual trace files of node’s bandwidth that can used to run simulations.

Table 5-13 reports on four pre-configured VM instances that a Cloud user can request to process cloudlets. The type of VM instance was set randomly to various configurations in order to have a better understanding of the allocation process provided by each tested VM allocation mechanism, similar to that explained in section 5.3.3. During the run time of the experiments, there were 700 instances of VM requested to process submitted cloudlets. The type of each requested VM instance was assigned randomly to the four VM types. Each Cloudlet was submitted by the cloudlet scheduler in DesktopCloudSim to a running VM instance to be processed. In the simulation, a time-shared scheduling policy [178] is employed to submit cloudlets to VM instances. This scheduling policy of cloudlets means that multiple cloudlets can be processed simultaneously. The estimated finish time (eft) required to execute a cloudlet *c* processed by a VM, *vm,* using time-shared scheduler is given as follows:

$$eft(c) = ct + \frac{cl}{capacity \times cores(c)} \tag{3}$$

where *eft(c)* denotes the estimated finish time of cloudlet *c*, *ct* denotes current simulation time, *cl* means the cloudlet length in MI, and *cores(c)* is the number of CPU cores needed to process cloudlet *c*. *Capacity* is the computing power in MIPS provided by a VM *vm*, calculated as follows:

$$capacity = \frac{\sum_{i=1}^{np} vm(i)_{cap}}{\max(\sum_{j=1}^{cloudlets} cores(j), np)} \tag{4}$$

where *np* is the number of cores that *vm*, $vm(i)_{cap}$ is the CPU power of core *i* of *vm,* and $\max(\sum_{j=1}^{cloudlets} cores(j), np)$ is the maximum cloudlet length of all submitted cloudlets to *vm.*

It is worth mentioning that the number of VMs remained unchanged in all experiments. Each VM instance was allocated to a PM using a VM allocation mechanism. The VM mechanism was changed from time to time to test the behaviour of each mechanism in the presence of node failure.

Table 5-13: VM Instance Types

VM Type	**VM Specification**	
	CPU (MIPS)	**RAM (GB)**
Micro	500	.633
Small	1000	1.7
Medium	2000	1.85
Large	2500	3.75

5.3.6 Evaluation Metrics

This section explains the metrics mentioned in section 4.5 from the perspective of the DesktopCloudSim simulation tool. The first metric is throughput, the number of successfully executed tasks to the total number of processed tasks. The throughput metric is given as follows:

$$throughput = \frac{\sum sc}{\sum pc} \tag{5}$$

The throughput of cloudlets is in percentage form in order to reflect the impact of node failure on the number of successfully executed tasks submitted by Cloud users, where *sc* denotes successful executed cloudlet and pc denotes a processed cloudlet. Processed cloudlets are any cloudlet that has been sent to a VM for processing. There is a chance that a cloudlet will not be successfully processed in which case the associated VM is destroyed as a result of its hosted PM failing during the execution time of the cloudlet. Power consumption was calculated in the experiment as follows:

$$power\ consumption = \sum_{i=1}^{n} p_u(pm_i) \tag{6}$$

Power consumption was calculated as total power consumption by nodes in a Cloud system per day, the 24-hour running time of an experiment set. The power consumption is given in kilowatt-hours (kWh) per day, where *n* stands for the number of PMs in a Desktop Cloud system, and p_u is the given power consumption of node *i* in the simulation run, depending on the utilisation *u* of the target PM i. Each PM has a listed power consumption according to its utilisation level. For example, a SuperMicro PM can consume about 0.223 kWh when its utilisation is 30%. The power consumption of a PM varies from one machine to another, depending in two factors: first, the type of the measured machine and, second, its utilisation level, detailed on the SPEC website as mentioned in section 5.3.3. Power consumption is given in the form of the total power consumed by PMs in a Desktop Cloud system for 24 hours, and the availability metric is given as follows:

$$availability = \sum_{i=1}^{n} \frac{avl(pm_i)}{CPU(pm_i)} \tag{7}$$

where $CPU(pm_i)$ is the total CPU power in MIPS for the PM with id *i*, *n* denotes the total number of PMs in a Desktop Cloud system, $avl(pm_i)$ is the current availability of a PM with id = *i* in MIPS. The current availability is given as follows:

$$avl(PM_i) = CPU(PM_i) - \sum_{j=1}^{m} CPU(vm_j) \tag{8}$$

where *m* means the number of hosted VMs to the PM pm_i, and $CPU(vm_j)$ is the computing power, in MIPS, of a VM *vm* with id j, hosted to the PM pm_i.

5.3.7 Experiment Methodology

The experiment was run 180 times, each simulating a day's (i.e. 24 hours) running of FTA nodes, as discussed in section 5.3.2, in a Desktop Cloud system for each VM allocation mechanism: once for a public Cloud and once for a private Cloud. Each simulation gave results on an hourly basis for each metric. In total there were 180 simulation runs × 4 VM mechanisms × 3 metrics × 2 Desktop Types, which provided 4320 records that were analysed in this experiment. There were two input data sets for the simulation tool DesktopCloudSim, as discussed in section 5.1.3. The first was the FTA files to simulate nodes in the infrastructure, along with failure event times in two scenarios. One scenario used NotreDame trace files to simulate a private Desktop Cloud, and the other simulated a public Desktop Cloud by using SETI@home FTA files. The second input data set was a workload of tasks submitted by users for processing in VMs, as mentioned in section 5.3.4.

The FTA files provided the numbers and IDs of nodes. However, the specifications of nodes were missing from the archive; as a result, they were set randomly for the PMs. The missing specifications were technical, such as CPU power, RAM and hard disk size. The type and specification of the PMs are mentioned in section 5.3.3. The number of requested VM instances was 700 instances, to be run for 24 hours. VM instances were classified into *micro, small, medium and large* VM types, as offered by Amazon EC2. Each VM instance received an equal series of tasks to process in a given workload.

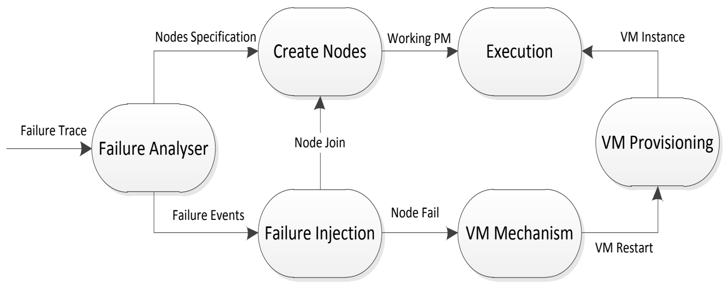

Figure 5-11: VM Mechanism Steps

If a node failed in the experiment, then all hosted VMs were destroyed, as Figure 5-11 depicts.

The destruction of a VM caused all running tasks on the VM to be set as ‘failing’. The lost VM was started again and allocated to another PM to process tasks. However, if there was a replica for the failing VM, this continued executing the tasks, as discussed in section 5.3.5. A new replica was instantiated and allocated to a PM. The simulation was run using Eclipse Software Development Kit V4.2.0 on a Mac i27 (CPU = 2.7 GHz Intel Core i5, 8 GB MHz DDR3) running OS X 10.9.4. The results were analysed using Microsoft Excel 2011, IBM SPSS Statistics v22 software and OriginLab Origin 8.

5.4 Baseline Experiments

This experiment compared Desktop Clouds to Traditional Clouds using three metrics: throughput; power consumption; and availability. The experiment was to answer the following research question:

What is the difference between Desktop Clouds and Traditional Clouds in terms of throughput, power consumption and availability?

To investigate the comparison in depth, the evaluation metric was studied 12 times, six for the Desktop Cloud and six for the Traditional Cloud. Each VM mechanism was used once for Desktop Cloud and once for the Traditional Cloud. The 12 sets are reported in Table 5-14. NotreDame Clouds represent private Clouds, both Desktop and Traditional Clouds, while SETI@home Clouds represent public Clouds, both Desktop and Traditional Clouds.

Table 5-14: Baseline Experiment Sets

VM Mechanism	NotreDame Scenario	SETI@home Scenario
FCFS	Desktop vs. Traditional	Desktop vs. Traditional
Greedy	Desktop vs. Traditional	Desktop vs. Traditional
RoundRobin	Desktop vs. Traditional	Desktop vs. Traditional

The reliability of Cloud nodes is 99.99% in Traditional Clouds, according to Vishwanath and Nagappan (2010). This high level of reliability leads to a high throughput level, close to 100%. The FCFS, Greedy and RoundRobin mechanisms are implemented as explained in section 3.2.1. None of these mechanisms, however, employ a replication technique.

5.4.1 FCFS VM Mechanism

The FCFS mechanism indicated that a mean of about 99.99% of submitted cloudlets was successfully executed in NotreDame Traditional Cloud, while it was about 82.66% in NotreDame Desktop Cloud. For SETI@home Clouds, the throughput was 99.99% in Traditional Clouds. The throughput value in SETI@home Desktop Cloud was about 82.04%. Throughput results are reported in Table 5-15 and Table 5-16.

Table 5-15: Descriptive Results, FCFS Mechanism, NotreDame Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	82.66%	82.2%	40.32	6.35	99.99%	99.99%	n/a	n/a
Power Consumption	533 kWh	538 kWh	867	29.45	523 kWh	530 kWh	826	28.74
Availability	85.03%	84.59%	4.21	2.05	85.29%	85.17%	4.46	2.11

Table 5-15 and Table 5-16 report power consumed by nodes in Desktop and Traditional NotreDame Clouds and SETI@home Clouds. For NotreDame Clouds, nodes consumed an average of 533 kWh per day, but an average of 523 kWh when the FCFS mechanism was used in NotreDame Traditional Cloud. The Kolmogorov-Smirnov Test of normality was conducted on the results of power consumption NotreDame Clouds. The test showed that the results were statistically significantly non-normal, $\mathrm{p} < .05$, therefore the non-parametric test for two related sample Wilcoxon signed ranks test was used. According to the test, nodes in NotreDame Desktop Cloud consumed statistically significantly more power (median = 538 kWh) than nodes in NotreDame Traditional Cloud (median = 530 kWh), $Z = -6.89, \mathrm{p} < .001$, at the 95% level of confidence. The case is different for the power consumption metric in SETI@home Desktop and Traditional Clouds because the results were normally distributed, $\mathrm{p} > .05$. The paired T-test showed that, on average, nodes within SETI@home Desktop Cloud consumed statistically significantly less power (mean = 507 kWh, SE = .86) than their counterparts in Traditional Clouds (mean = 518 kWh, SE = .49), $t(179) = -12.04, \mathrm{p} < .001$, at the 95% level of confidence.

Table 5-16: Descriptive Results, FCFS Mechanism, SETI@home Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	82.04%	83.28%	20.23	4.5	99.99%	99.99%	n/a	n/a
Power Consumption	507 kWh	506 kWh	131.85	11.48	518 kWh	519 kWh	43.81	6.62
Availability	91.81%	91.8%	.05	.23	93.42%	93.42%	.03	.16

Nodes in the NotreDame Traditional Cloud were statistically significantly more available to host new VMs (median = 85.17%) than nodes in the Desktop Cloud (median = 84.59%), according to Wilcoxon test, $Z = -4.96, \mathrm{p} < .001$, at the 95% level of confidence. Results of the availability of nodes in SETI@home were statistically significantly normally distributed. Nodes in SETI@home Traditional Cloud (mean = 93.42%, SE = .01) were more available than in Desktop Cloud (mean = 91.81%, SE = .02), $t(179) = -32.57, \mathrm{p} < .001$, at the 95% level of confidence.

5.4.2 Greedy VM Mechanism

The mean and the median of throughput for NotreDame Desktop Cloud were 92.47% and 93.1% when the Greedy mechanism was employed, while the throughput for NotreDame Traditional Cloud was close to 100%, as Table 5-17 shows.

Table 5-17: Descriptive Results, Greedy Mechanism, NotreDame Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	92.47%	93.1%	18.34	4.28	99.99%	99.99%	n/a	n/a
Power Consumption	638 kWh	641 kWh	738	27.16	685 kWh	690 kWh	796	28.21
Availability	86.22%	86.23%	3.01	1.76	86.57%	86.55%	3.2	1.79

Table 5-18 shows that the average throughput of SETI@home Desktop Cloud was about 81.80%. This demonstrated that the average throughput of Desktop Clouds was about 7.52% less than Traditional Clouds for private Clouds and about 18.19% less for public Clouds.

According to the Kolmogorov-Smirnov test of normality, the results of power consumption results when the Greedy mechanism was employed for both Desktop and Traditional NotreDame Clouds were significantly non-normal, $p < .05$. Therefore, Wilcoxon test was again used to compare the power consumed by nodes in both NotreDame Desktop and Traditional Clouds and showed that the Desktop Cloud consumed statistically significantly less power (median = 641 kWh) than the Traditional Cloud (median = 690 kWh), $Z = -11.6, p < .001$, at the 95% level of confidence. The Wilcoxon test was also used to evaluate power consumption in SETI@home Clouds because results were significantly non-normal, $p < .05$, showing that nodes in SETI@home Desktop Cloud consumed statistically significantly less power (median = 696 kWh) than nodes in SETI@home Traditional Cloud (median = 835 kWh), $Z = -11.64, p < .001$, at the 95% level of confidence.

Table 5-18: Descriptive Results, Greedy Mechanism, SETI@home Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	81.80%	81.93%	16.1	4.01	99.99%	99.99%	n/a	n/a
Power Consumption	694 kWh	696 kWh	615	24.79	834 kWh	835 kWh	25.9	5.01
Availability	92.59%	92.6%	.1	.31	93.42%	93.42%	.03	.16

The results of the availability metric for NotreDame Clouds were non-normal, as the Kolmogorov-Smirnov test showed, $p < .05$. Therefore, the Wilcoxon test was applied to compare the availability of nodes and showed that nodes of NotreDame Traditional Clouds were statistically significantly more available to host new VMs (median = 86.55%) than their counterparts in Desktop Clouds (median = 86.23%), $Z = -6.57, \mathrm{p} < .001$, at the 95% level of confidence. It is worth mentioning that the difference was quite small, at less than .5%. The results of node availability for Desktop and Traditional SETI@home Clouds were normally distributed, according to the Kolmogorov-Smirnov test, $\mathrm{p} > .05$. Therefore, the paired T-test was conducted to compare the availability notation. The test showed that there was a statistically significant difference between availability of nodes in Desktop (mean = 92.59%, SE = .02) and Traditional Clouds (mean = 93.42%, SE = .01), $t(179) = -32.57, \mathrm{p} < .001$, at the 95% level of confidence.

5.4.3 RoundRobin VM Mechanism

The average throughputs of the NotreDame and SETI@home Desktop Clouds were about 89.14% and about 80.45% when the RoundRobin mechanism was employed, as reported in Table 5-19 and Table 5-20. The throughput of NotreDame and SETI@home Traditional Clouds remained close to 100%, similar to that reported when the FCFS and Greedy mechanisms were used.

Table 5-19: Descriptive Results, RoundRobin Mechanism, NotreDame Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	89.14%	89%	16.47	4.06	99.99%	99.99%	n/a	n/a
Power Consumption	1884 kWh	1883 kWh	22236	149012	2000 kWh	1999 kWh	14513	120.47
Availability	81.98%	81.91%	2.55	1.6	81.63%	81.58%	2.48	1.57

The Kolmogorov-Smirnov test showed that the power consumption distribution of the Desktop and Traditional NotreDame Clouds was significantly non-normal, $\mathrm{p} < .05$, while normally distributed, $\mathrm{p} > .05$. Therefore, the Wilcoxon test was used to compare the power consumed by nodes in NotreDame Clouds and the paired T-test was conducted to study the power consumption results in SETI@home Clouds. For the NotreDame Desktop Cloud, the Wilcoxon test showed that the nodes in the Desktop Cloud consumed statistically significantly less power (median = 1883 kWh) than nodes in Traditional Cloud (median = 1999 kWh), $Z = -11.55, \mathrm{p} < .001$, at the 95% level of confidence. For SETI@home, paired T-test showed that the average (mean = 2216 kWh) power consumed by nodes Desktop Cloud was less than the power consumed by Traditional Clouds' nodes (mean = 2387 kWh). However, the T-test showed that the results were not statistically significant, because the critical value $\mathrm{p} > .05$.

Table 5-20: Descriptive Results, RoundRobin Mechanism, SETI@home Clouds

	Desktop Clouds				Traditional Clouds			
Metric	**Mean**	**Median**	**Var**	**Std Dev**	**Mean**	**Median**	**Var**	**Std Dev**
Throughput	80.45%	81.04%	16.11	4.01	99.99%	99.99%	n/a	n/a
Power Consumption	2216 kWh	2215 kWh	2184.53	46.74	2387 kWh	2387 kWh	866.41	29.44
Availability	88.83%	88.84%	.1	.31	90.32%	90.3%	.05	.21

The results of availability of NotreDame Clouds' results did not follow a normal distribution, therefore the Wilcoxon test was used for the evaluation. The test showed that nodes were statistically significantly more available in the NotreDame Desktop Cloud (median = 81.91%) than the NotreDame Traditional Cloud (median = 81.58%), $Z = -6.43, \mathrm{p} < .001$, at the 95% level of confidence. However, the difference between the availability of Desktop and Traditional nodes was very small, at .3%. The situation was the opposite for the SETI@home Clouds because fewer computing nodes in the Desktop Cloud were ready to host new VMs (mean = 88.83%, SE = .02) than nodes in the Traditional Cloud (mean = 90.32%, SE = .02), $t(179) = -53.24, \mathrm{p} < .001$, as the T-test revealed at the 95% level of confidence.

5.4.4 Discussion

The experiment was conducted to study the difference in outcomes of Desktop Cloud systems and Traditional Cloud systems in terms of throughput, power consumption and availability. The FCFS, Greedy and RoundRobin VM allocation mechanisms in the literature were applied individually to four types of Cloud systems: NotreDame Desktop Cloud; NotreDame Traditional Cloud; SETI@home Desktop Cloud; and SETI@home Traditional Cloud. The experiment demonstrated that the throughput of a Cloud system is highly impacted by node failure. In the NotreDame Desktop Cloud, the private Cloud scenario, at least 7% of submitted tasks were lost when the Greedy mechanism was employed, due of node failure, whereas this measure was as little as .01% in the NotreDame Traditional Cloud. The number of missing tasks actually increased to about 18% in the SETI@home Desktop Cloud, remaining consistent in the SETI@home Traditional Cloud. The throughput of Traditional Clouds remained unchanged, as the experiment assumed no node failures would occur in Traditional Clouds since it has been demonstrated that the reliability of nodes in a Traditional Cloud system over a single day's running time is about 99.99%.

The experiment showed that the difference between nodes in Desktop Clouds and Traditional Clouds in terms of power consumption was limited, even in the presence of node failure events. For example, using the FCFS mechanism, the difference was only about 10 kWh between NotreDame Desktop and Traditional Clouds, and between SETI@home Desktop and Traditional Clouds. The power consumption when the Greedy mechanism was employed was similar to NotreDame Desktop and Traditional Clouds, at just 50 kWh average per running day. Such figures show that power consumption in both Desktop and Traditional Clouds is not affected by node failure, because power consumption is directly affected by number of requested VM instances and utilisation of hosting nodes.

The availability of nodes to host new VM instances is affected by node failure, as there are fewer nodes to host VM instances. The experiment confirmed this by comparing the availability of nodes in Desktop Cloud systems with Traditional Cloud systems; for example, the availability of nodes in SETI@home Desktop Cloud when the RoundRobin mechanism was used was 2% less than in the SETI@home Traditional Cloud. However, the impact of node failure is quite small because the VM instances play the key role in the availability metric.

The results demonstrate that throughput can be clearly affected by node failures in Desktop Clouds no matter which VM was employed. Therefore, it is crucial to develop a fault-tolerant VM mechanism that can improve the throughput of Desktop Cloud systems. According to the experiment and results, it can be said that DesktopCloudSim is able to simulate node failures in Cloud systems and thus the tool is capable of simulating Desktop Cloud systems.

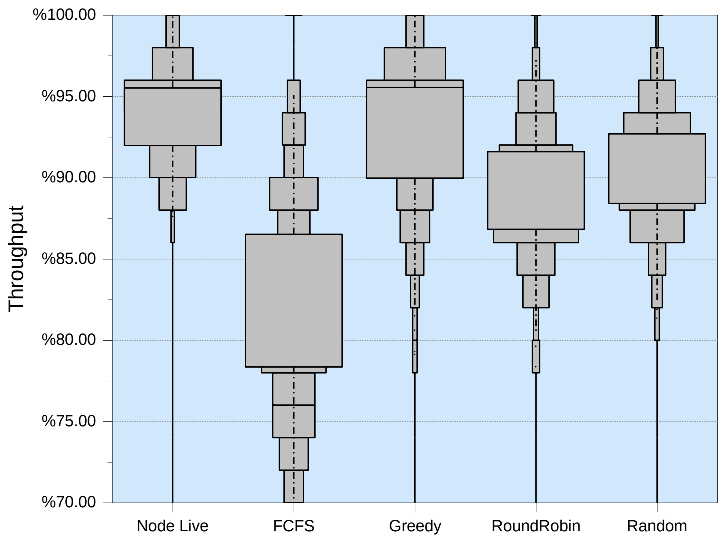

Figure 5-12: NotreDame Summary Results

Figure 5-12 and Figure 5-13 summary the range and average throughput results of the used mechanisms for NotreDame and SETI@home systems. “Node Alive” in the graph means the number of nodes that remain live during the run time of the experiment out of the total number of running PMs. The figures demonstrate the validity of the simulation tools to produce valid results when node failure events are applied. For NotreDame system, augmenting the simulation tool with about 90% – 94% remaining alive yields results that are almost similar for each mechanisms in terms of throughput. FCFS mechanism was slightly different by about 7% due to the fact that FCFS mechanism is expected to produce poor throughput according the literature (section 3.2.1). The same applies also for the SETI@home case where the throughput output was affected negatively because the failure rate increased. Therefore, it can be concluded that DesktopCloudSim tool simulates the effect of node failures as it is expected.

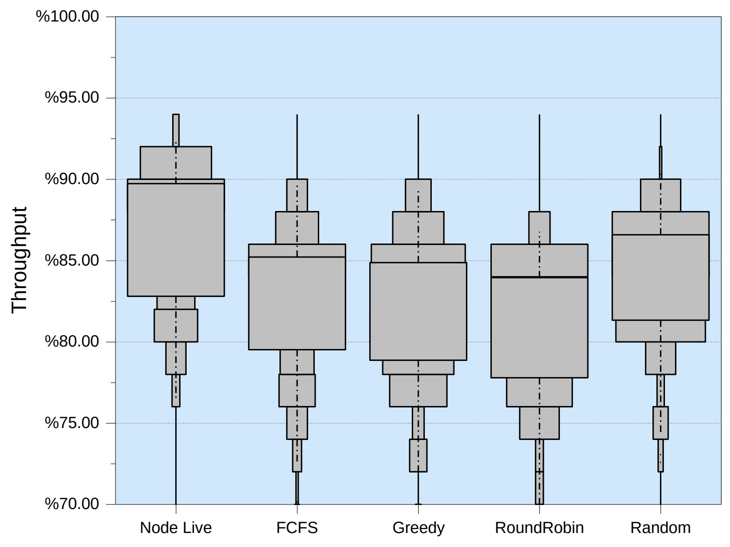

Figure 5-13: SETI@home Summary Results

5.5 Summary

This chapter explained the approach to be used to conduct experiments in this study. It showed that simulation can be an effective method of investigating research problems and evaluating proposed solutions. There is no real Desktop Cloud system, however, so using simulation to model these systems is justified. CloudSim is a well-known tool to simulate Cloud systems, but cannot simulate node failure although this feature is essential in this study to simulate Desktop Cloud systems. Therefore, the DesktopCloudSim tool was developed as an extension to CloudSim in order to simulate Desktop Clouds, and a brief explanation of the statistical analysis tests to be conducted in analysing results was presented.

The chapter also explained the design of experiments conducted in this study. Node failure events in DesktopCloudSim are simulated using real traces collected from the FTA archive. This provides failure events of nodes in NotreDame and SETI@home Desktop Grid systems. The failure data of NotreDame and SETI@home were studied in order to demonstrate that in such resources it is the norm rather than the exception. The specification of nodes in each simulated Desktop Cloud system, the NotreDame and SETI@home Desktop Cloud, was reported. In addition, an explanation followed on the workload of tasks submitted to a Desktop Cloud system.

Table 5-21: Private Cloud Systems

	Desktop Clouds			Traditional Clouds		
Metric	**FCFS**	**Greedy**	**RoundRobin**	**FCFS**	**Greedy**	**RoundRobin**
Throughput	82.66%	92.47%	89.14%	99.99%	99.99%	99.99%
Power Consumption	533 kWh	638 kWh	1884 kWh	523 kWh	685 kWh	2000 kWh
Availability	85.03%	86.22%	81.98%	85.29%	86.57%	81.63%

Table 5-22: Public Cloud Systems

	Desktop Clouds			Traditional Clouds		
Metric	**FCFS**	**Greedy**	**RoundRobin**	**FCFS**	**Greedy**	**RoundRobin**
Throughput	82.04%	81.80%	80.45%	99.99%	99.99%	99.99%
Power Consumption	507 kWh	694 kWh	2216 kWh	518 kWh	834 kWh	2387 kWh
Availability	91.81%	92.59%	88.83%	93.42%	93.42%	90.32%

Baseline experiments concluded the chapter to show that DesktopCloudSim may be used to investigate the difference between Desktop Clouds and Traditional Clouds in the light of the evaluation metrics: the throughput outcome of the submitted workload; the energy consumed by nodes; and the availability of nodes to host new VM requests. These showed that throughput is greatly influenced by node failure, but that its influence on power consumption and availability is limited. The results obtained for a private Desktop Cloud system (NotreDame system) are reported in Table 5-21, while Table 5-22 reports the results of a public Desktop Cloud system (SETI@home system).

The next chapter presents the evaluation experiments conducted based on the methodology proposed in this chapter.

Chapter 6: Experiment Results

Section 5.3.2 presented an analysis of node failure events occurring in the Desktop Grid nodes of NotreDame and SETI@home systems in order to demonstrate the probability of node failure in a Desktop Cloud system. Section 3.2 Chapter 3: reviewed the VM allocation techniques employed in Cloud systems, mainly Traditional Clouds. Section 3.2.5 showed that these VM mechanisms cannot handle node failure in Desktop Clouds, which explains the need to develop one that is tolerant of this problem. Section 4.3 proposed a new VM mechanism, the FT mechanism that recognises node failure. However, it seemed to consume a great amount of power, therefore section 4.4 proposed a utilisation-migration technique to help reduce the power consumption of nodes in Desktop Cloud systems when the FT mechanism is employed.

The main objective of this chapter is to examine through experiment three research hypotheses:

> H1: *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds*
>
> H2: *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud system*
>
> H3: *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with a decrease of 2% in the throughput outcome*

Section 5.3 explained how these experiments were conducted with the DesktopCloudSim simulation tool. The results of each experiment are described and analysed in a separate section. The first experiment demonstrates how node failure affects the throughput of Desktop Clouds when using VM mechanisms (FCFS, Greedy, RoundRobin and Random mechanisms) gathered from the literature. The second experiment demonstrates the effectiveness of employing the FT VM allocation mechanism in terms of improving throughput. The third section demonstrates how the UBMP can reduce the power consumption of nodes in Desktop Clouds when the FT mechanism is employed. Finally, a summary of the findings of this chapter is presented.

6.1 Experiment I: The Impact of Node Failure

Four VM allocation mechanisms – FCFS, Greedy, RoundRobin and Random – were compared using the evaluation metrics (discussed in section 4.5) in Desktop Clouds, specifically in instances of failure. The experiment was conducted in order to study the impact of node failure on throughput, power consumption and availability in Desktop Clouds. This experiment was conducted to test Hypothesis H1:

> *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds*

This experiment aimed to answer the following research questions:

> *Q1.1- What is the impact, if any, of node failure on throughput?*
>
> *Q1.2- Which VM allocation mechanism yields the highest throughput of the tested mechanisms in Desktop Clouds?*
>
> *Q1.3- Which VM allocation mechanism consumes the least power of the tested mechanisms in Desktop Clouds?*
>
> *Q1.4- Which VM allocation mechanism yields the best availability of the tested mechanisms in Desktop Clouds?*

Each evaluation metric is analysed separately in a different section to report the results of both private and public Desktop Cloud scenarios (as discussed in section 5.3). The private Desktop Cloud is represented by the NotreDame data set and the public Desktop Cloud is represented by SETI@home data set. The following subsection describes the methodology of this experiment. The second section describes and analyses the results of the experiment. The results are described for each evaluation metric separately. Finally, a discussion of the findings of this experiment is presented in the last subsection.

6.1.1 Results

The results were analysed in three subsections, according to the evaluation metrics: throughput, power consumption and availability. Throughput was calculated as in equation (5). Power consumption was calculated as in equation (6). Availability was calculated as in equation (7).

6.1.1.1 Throughput

Table 6-1 shows a summary of the descriptive results obtained when measuring the throughput metric for each VM allocation mechanism for the NotreDame Cloud. The Kolmogorov-Smirnov (K-S) test of normality showed that the normality assumption was not satisfied, because the results of FCFS and Greedy mechanisms were statistically significantly non-normal: P < .05. Therefore, the non-parametric test of Friedman's ANOVA was used to test which mechanism would yield the highest throughput. This confirmed that throughput varied statistically significantly from mechanism to mechanism:$X_F^2(3) = 397.14, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-1.

Table 6-1: Impact of Node Failure on Throughput Metric, NotreDame Cloud

Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
FCFS	82.66	82.2	40.32	6.35	.47	P = .034
Greedy	92.47	93.1	18.34	4.28	.32	P< .001
RoundRobin	89.14	89	16.47	4.06	.30	P = .2
Random	90.63	90.63	13.15	3.63	.27	P = .2

Six post-hoc analyses with Wilcoxon pairwise comparison tests were conducted to test if each mechanism was statistically significantly different from others. Note that six tests were required to compare six pairs of mechanisms listed in Table 6-2. The level of significance was adjusted to be 0.008 using the Bonferroni correction method [169], because six post-hoc tests were required (.05/6 ≈ .008). Table 6-2 shows a statistically significant difference between each mechanism and its counterparts. Therefore, it can be said that the Greedy mechanism gave the highest throughput statistically significantly, since it has the median with the highest value at93.1%.

Table 6-2: Pairwise Comparisons: Impact of Node Failure on Throughput Metric, NotreDame Cloud

Pairwise Mechanism Comparison	Wilcoxon Test
FCFS vs. Greedy	P < .008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

The aforementioned VM mechanisms were also tested when applied to a public Cloud using the SETI@home dataset. Table 6-3 reports on the mean, median, variance (Var), standard deviation (Std Dev), standard error (SE) and the K-S test. According to the K-S normality test, the results did not follow a normal distribution: P < .05. Friedman's ANOVA test indicated that there was a significant difference between the tested mechanism, $X_F^2(3) = 266.24, P < .001$, at the 95% level of confidence.

Table 6-3: Impact of Node Failure on Throughput Metric, SETI@home Cloud

Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
FCFS	82.04	83.28	20.23	4.5	.34	P < .001
Greedy	81.80	81.93	16.1	4.01	.3	P = .2
RoundRobin	80.45	81.04	16.11	4.01	.3	P = .004
Random	83.72	84.33	12.83	3.58	.27	P = .006

Similar to the results with the NotreDame Desktop Cloud, six pairwise Wilcoxon tests were applied, as shown in Table 6-4. The tests showed that there were statistically significant differences between all pairs except the FCFS vs. Greedy mechanisms, where they indicated that the difference was not statistically significant: P > .008. However, this does not affect the overall finding because of the results for the Random mechanism, which was statistically significantly different from the others. Therefore, it can be said that the Random mechanism gave the highest throughput statistically significantly (median = 84.33%).

Table 6-4: Pairwise Comparisons, Impact of Node Failure on Throughput Metric, SETI@home Cloud

Pairwise Mechanism Comparison	Wilcoxon Test
FCFS vs. Greedy	P >.008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

6.1.1.2 Power Consumption

This section evaluates the FCFS, Greedy, RoundRobin and Random mechanisms in terms of power consumed by nodes in Desktop Clouds. The mechanisms were implemented in a private Cloud using the NotreDame data set in order to be tested. Friedman's ANOVA test, which was applied to the power consumption results, showed that there was a statistically significant difference between the tested mechanisms:$X_F^2(3) = 540, P < .001$, at the 95% level of confidence. Friedman's ANOVA test was selected because the power consumption results were not normally distributed, since the critical value (P-value) was <.05 for the

FCFS and the Greedy mechanisms. Table 6-5 reports on the mean, median, variance (Var), standard deviation (Std Dev), standard error (SE) and the K-S test of normality.

Table 6-5: Impact of Node Failure on Power Consumption Metric, NotreDame Cloud

Mechanism	**Mean (kWh)**	**Median (kWh)**	**Var**	**Std Dev**	**SE**	**K-S Test**
FCFS	533	538	867	29.45	2.2	P < .001
Greedy	638	641	738	27.16	2.03	P = .005
RoundRobin	1884	1883	22237	149	11.12	P = .2
Random	1368	1369	6946	83.35	6.12	P = .2

Table 6-6 summarises the results of pairwise comparison of the Wilcoxon test conducted to study which mechanism consumed the least power. This showed that there was a statistically significant difference between each pair mechanism. Therefore, it can be said that when the FCFS mechanism was employed, the nodes consumed the least power statistically significantly (the median is 538 kWh).

Table 6-6: Pairwise Comparisons, Impact of Node Failure on Power Consumption Metric, NotreDame Cloud

Pairwise Mechanism Comparison	**Wilcoxon Test**
FCFS vs. Greedy	P < .008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

The same mechanisms were evaluated in terms of power consumption when the mechanisms were employed in a public Cloud using SETI@home data set. Table 6-7 reports on the mean, median, variance and K-S test. Both the Greedy and Random mechanisms were statistically significantly non-normally distributed: P <.05. Therefore, the non-parametric Friedman's ANOVA test was applied to test if there was a significant difference between the results. The test showed that there was indeed a statistically significant difference: $X_F^2(3) = 540, P < .001$, at the 95% level of confidence.

Table 6-7: Impact of Node Failure on Power Consumption Metric, SETI@home Cloud

Mechanism	**Mean (kWh)**	**Median (kWh)**	**Var**	**Std Dev**	**SE**	**K-S Test**
FCFS	507	506	131.85	11.48	.86	P = .2
Greedy	694	696	614.7	24.79	1.85	P = .015
RoundRobin	2217	2215	2185	46.74	3.84	P = .2
Random	1533	1534	3263	57.12	4.26	P = .002

Pairwise comparisons tests were conducted to find out which mechanism consumed the least power, as shown in Table 6-8. There were statistically significant differences between the tested pairs of mechanisms. Therefore, it can be said that, when the FCFS mechanism was employed, the nodes consumed the least power statistically significantly (the median is 506 kWh).

Table 6-8: Pairwise Comparisons, Impact of Node Failure on Power Consumption Metric, SETI@home Cloud

Pairwise Mechanism Comparison	**Wilcoxon Test**
FCFS vs. Greedy	P < .008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

6.1.1.3 Availability

Table 6-9 shows a summary of the descriptive results obtained when measuring the availability metric for each VM allocation metric in NotreDame Cloud. According to the K-S test of normality, the results did not follow a normal distribution: P < .05. The non-parametric test, Friedman’s ANOVA, was used to test if there was a statistically significant difference between the results. It confirmed that availability varied statistically significantly from mechanism to mechanism: $X_F^2(3) = 510.78\,, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-9.

Table 6-9: Impact of Node Failure on Availability Metric, NotreDame Cloud

Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
FCFS	85.03	84.59	4.21	2.05	.15	P < .001
Greedy	86.22	86.23	3.09	1.76	.13	P < .001
RoundRobin	81.98	81.91	2.44	1.6	.12	P < .001
Random	80.69	80.55	3.5	1.87	.14	P < .001

Six Wilcoxon pairwise comparison tests were used to find out if there was a statistically significant difference between the Greedy mechanism, with the highest availability, and the other VM mechanisms. The level of significance was set to 0.008 using the Bonferroni correction method (.05/6 ≈ .008). Table 6-10 shows that there was a statistically significant difference between each pair of VM mechanisms. Therefore, it can be said that Greedy mechanism outperformed other mechanisms statistically significantly in terms of availability by looking at the median (86.22%).

Table 6-10: Pairwise, Impact of Node Failure on Availability Metric, NotreDame Cloud

Pairwise Mechanism Comparison	Wilcoxon Test
FCFS vs. Greedy	P < .008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

The availability of nodes was tested in a public Cloud using the SETI@home dataset when the FCFS, Greedy, RoundRobin and Random mechanisms were employed. The K-S test of normality showed that the results of mechanisms were normally distributed: P > .05, as in Table 6-11. Therefore, the repeated measure ANOVA test[169] was used to study the effect of the mechanisms on the availability metric for SETI@home Cloud. Mauchly's test [169] indicated that the assumption of sphericity had been violated: $x^2(5) = 58.57, p < .05$. Therefore, the degree of freedom was corrected by the Greenhouse-Geisser [169]estimates of sphericity ($\varepsilon = .82$).The test showed that the availability of nodes in the Clouds was affected statistically significantly by the employed VM allocation mechanism: $\mathrm{F}(2.45, 438.65) = 8265.29, \mathrm{p} < .05$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-11.

Table 6-11: Impact of Node Failure on Availability Metric, SETI@home Cloud

Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
FCFS	91.81	91.8	.06	.23	.17	P = .2
Greedy	92.59	92.6	.1	.31	.02	P = .2
RoundRobin	88.83	88.84	.1	.31	.02	P = .2
Random	88.67	88.65	.12	.34	.03	P = .2

The repeated measure ANOVA test showed that the availability varied statistically significantly. Six pairwise comparisons using the paired T-test [169] were conducted. The level of significance was adjusted to 0.008, using the Bonferroni correction. The results of post-hoc paired T-test tests showed that there were statistically significant differences between node availability for each VM mechanism, as shown in Table 6-12. Therefore, it can be said that the Greedy mechanism has the highest availability of nodes, statistically significantly, when employed in a public Cloud.

Table 6-12: Pairwise Comparisons, Impact of Node Failure on Availability Metric, SETI@home Cloud

Pairwise Mechanism Comparison	Paired T-Test
FCFS vs. Greedy	P < .008
FCFS vs. RoundRobin	P < .008
FCFS vs. Random	P < .008
Greedy vs. RoundRobin	P < .008
Greedy vs. Random	P < .008
RoundRobin vs. Random	P < .008

6.1.2 Discussion

Table 6-13 summarises the findings of the experiment conducted to compare the behaviour of four VM mechanisms: the FCFS; Greedy; RoundRobin; and Random mechanisms. This was in view of three metrics: throughput; power consumption; and availability. The Greedy mechanism was the mechanism that yielded the highest throughput level in a private Cloud, and the Random mechanism when used in a public Cloud. In terms of power consumption, the FCFS mechanism came first in both the private and public Clouds. The Greedy mechanism was the best in both private and public Clouds in terms of node availability.

Table 6-13: Summary of Impact of Node Failure Using Evaluation Metrics

	VM Mechanism	
Metric	**Private Cloud**	**Public Cloud**
Throughput	Greedy	Random
Power Consumption	FCFS	FCFS
Availability	Greedy	Greedy

In terms of throughput, the FCFS mechanism gave very low values, 82% on average. Such figures mean that 18% of submitted tasks were lost due to node failure in the NotreDame Desktop Cloud. The reason is that the FCFS mechanism allocates many VMs to a single PM: thus, if the PM fails, the number of lost VMs is quite high. In contrast, the Greedy mechanism gave the highest average value, at 92% on average. The Random and RoundRobin VM mechanisms gave throughput results of an average 91% and 89% for the NotreDame and the Desktop Cloud, respectively. On average, the percentage failure on the NotreDame Desktop Cloud was 6%, as mentioned in section 5.3.2.

It is showed in Figure 6-2 the range of throughput metric results for each VM mechanism. According to the figure, the FCFS mechanism yielded a minimum value of only 70% throughput, whereas minimum values for Greedy, RoundRobin and Random mechanisms were 80%. Note that the small dots in the figure represent the actual value of throughput obtained in the experiment.

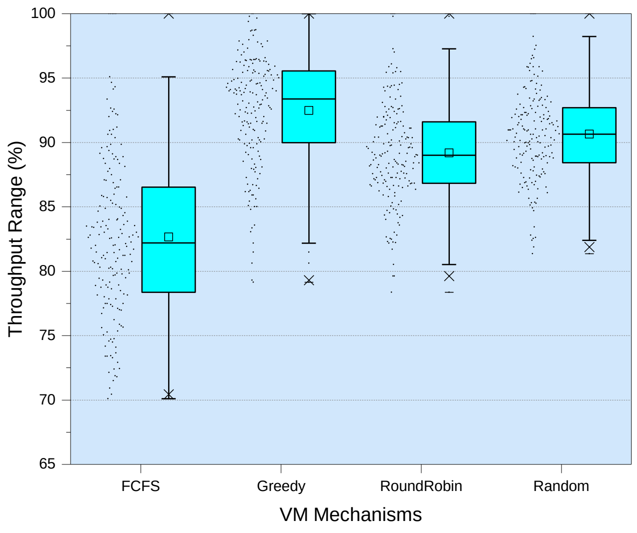

Figure 6-1: Throughput Range for VM Mechanisms for NotreDame System

Interestingly, the Random mechanism gave the best results for the SETI@home Desktop Cloud, at 84% on average, while the Greedy mechanism was 82%. Average throughput dropped by 10% from the NotreDame Desktop Cloud to the SETI@home Desktop Cloud, as the average node failure for the NotreDame is 6% per hour compared to 14% for the

SETI@home Desktop Cloud. Another interesting finding is that the average throughput of the Greedy, RoundRobin and Random VM mechanisms decreased by 9–11% from the NotreDame Desktop Cloud level when employed in the SETI@home Desktop Cloud. The FCFS mechanism also decreased, but by a very small amount(less than .2%). The explanation is that the FCFS was not affected by node failure figures as long it was above 6% (in the NotreDame Desktop Cloud case) and below 14% (in the SETI@home Desktop Cloud case). 7% of submitted tasks were lost when node failure was low (the NotreDame case), compared to 16% when it was high (the SETI@home case). Such lost tasks figures emphasise the need to design a VM allocation mechanism that is tolerant of node failure; furthermore, Figure 6-2 shows that the range of throughput outcomes for the evaluated mechanisms is quite small. The throughput of the Random mechanism, the best VM mechanism in SETI@home, drop only to just below 75%.

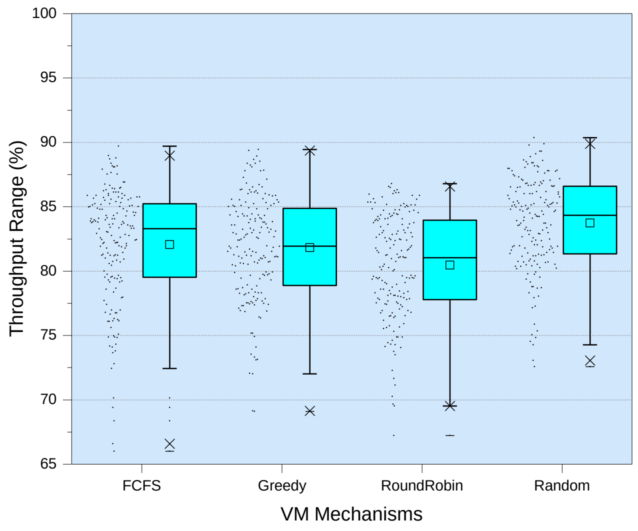

Figure 6-2: Throughput Range for VM Mechanisms for SETI@home System

The power consumed by nodes seems to be consistent across both the NotreDame and the SETI@home Desktop Cloud scenarios: The FCFS mechanism consumed the least power in both Desktop Clouds, at on average 533 kWh and 507 kWh, respectively. The Greedy mechanism came a close second, with figures of 638 kWh and 694 kWh. These figures are low compared to those of the Random and RoundRobin mechanisms, because the FCFS and the Greedy mechanisms improved utilisation by allocating as many VMs as possible to the same PM. By contrast, the RoundRobin mechanism distributed VMs to as many as possible PMs to balance the load. This meant that many PMs hosted VMs at low utilisation levels. The RoundRobin mechanism consumed more power in the SETI@home Desktop Cloud (2217 kWh) than in the NotreDame (1884 kWh), because the number of physical nodes was greater in the former.

Similarly, the results of the availability metric were consistent across both the NotreDame and the SETI@home Desktop Clouds, since the Greedy mechanism had the highest resource availability. It is worth mentioning that there was no great difference between the best mechanism, the Greedy mechanism, and the worst, the Random mechanism, at only 5%. Such a minor difference for both private and public Desktop Clouds may mean that the mechanism employed plays only a small role in improving resource availability.

Table 6-14: Evaluation Metrics for Desktop Cloud Systems

	Private System				**Public System**			
Metric	**FCFS**	**Greedy**	**RoundRobin**	**Random**	**FCFS**	**Greedy**	**RoundRobin**	**Random**
Throughput	82.66%	92.47%	89.14%	90.63%	82.04%	81.80%	80.45%	83.72%
Power Consumption	533 kWh	638 kWh	1884 kWh	1368 kWh	507 kWh	694 kWh	2217 kWh	1533 kWh
Availability	85.03%	86.22%	81.98%	80.69%	91.81%	92.59%	88.83%	88.67%

Table 6-14 presents a summary of results obtained from this experiment. The answer of Question *Q1.1* is that node failure reduced throughput in both private and public Desktop Cloud systems, no matter which mechanism was employed. In answer to Question *Q1.2,* it was found that Greedy VM allocation is the mechanism yielding the highest throughput when employed in a private Desktop Cloud system. The FCFS mechanism was the answer to Question *Q1.3,* being the one that consumed the least power. The Greedy mechanism was indeed the answer to *Q1.4,* as it yielded the greatest availability of PMs, when employed in private and public Desktop Cloud systems. By answering the above research questions the experiment demonstrated that metrics are able to evaluate the impact of node failure on a Desktop Cloud system and that there is a need to develop a novel VM mechanism that can cope with its issue of node failure.

6.2 Experiment II: Evaluation of the FT Mechanism

The first experiment showed that 7% of workload in a private Desktop Cloud and 16% of that in a public Desktop Cloud can be lost as a result of node failure. Such figures emphasise the need to implement a fault-tolerant VM allocation mechanism. The second experiment was conducted to evaluate the FT VM mechanism proposed in section 4.3 using the evaluation metrics: throughput; power consumption; and availability. This experiment compared the FT mechanism with that which gave the best results in each evaluation metric in the previous experiment. The experiment tested Hypothesis H2:

Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud system

The experiment aimed to answer the following questions:

Q2.1 - What is the impact of employing the FT mechanism on the power consumed by nodes with a Desktop Cloud system?

Q2.2 - What is the impact of employing the FT mechanism on the availability of Desktop Clouds' nodes?

Each evaluation metric is analysed in a separate section. Each reports the results of both a private Desktop Cloud, represented by the scenario of the NotreDame data set, and a public Desktop Cloud, represented by the scenario of the SETI@home data set. The following subsection describes the methodology of this experiment, while the next describes and analyses the results of the experiment. The results are reported for each evaluation metric separately. The last subsection discusses the findings of this experiment.

6.2.1 Results

Similar to the previous results, each evaluation metric is analysed in a separate subsection. Each section reports the results of both private and public Desktop Clouds. Throughput was calculated as given in equation (5). Power consumption was calculated as in equation (6). Availability was calculated as in equation (7).

6.2.1.1 Throughput

When the Greedy mechanism was employed the NotreDame Desktop Cloud yielded a higher throughput than with the FCFS, RoundRobin and Random VM mechanisms, as shown in section 6.1.1.1. Therefore, this was selected to be compared with the FT mechanism in order to evaluate its efficiency. According to the K-S test, the results of the mechanisms were not statistically significantly normal: $\mathrm{p} < .05.$ Therefore, the Wilcoxon test was applied to compare their median throughput. The test showed that the FT mechanism yielded a throughput level (median = 99.99%) that was better, statistically significantly, than the level of the Greedy mechanism (median = 93.37%) by 6%: $Z = -11.57, \mathrm{p} < .001$, at the 95% level of confidence. The mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-15.

Table 6-15: Throughput Metric for FT Mechanism

Cloud Type	Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
Private Cloud	Greedy	92.47	93.37	18.34	4.28	.32	P < .001
	FT	99.89	99.99	.09	.29	.02	P < .001
Public Cloud	Random	83.72	84.33	12.83	3.58	.27	P = .006
	FT	99.88	99.99	.1	.32	.02	P < .001

For the public Cloud, the Random mechanism was selected for comparison with the FT mechanism because, according to section 6.1.1.1, it gave the highest throughput results. The K-S normality test showed that the normality assumption of results was violated: P < .05. Therefore, the Wilcoxon test was conducted to establish whether there was a statistically significant difference between the results of the Random and the FT mechanisms. The test showed that there was indeed a statistically significant difference between them: $Z = -11.64, \mathrm{p} < .001$, at the 95% level of confidence. The mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-15. Therefore, it can be concluded that the throughput of both private and public Desktop Clouds improved statistically significantly when the FT VM allocation mechanism was employed.

6.2.1.2 Power Consumption

This section compares the results of power consumed by nodes in private and public Desktop Clouds under the FT mechanism and the FCFS mechanism. The FCFS mechanism was selected for comparison because it consumed the least power in private and public Clouds, as section6.1.1.2 showed. The mean, median, variance (Var.), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-16 for the two Cloud types. According to the Wilcoxon test, nodes in private Clouds consumed (median = 533 kWh) less power, statistically significantly, when the FCFS mechanism was used than when the FT mechanism (median = 1108 kWh) was used: $Z = -11.64, \mathrm{p} < .001$, at the 95% level of confidence. Likewise, nodes in public Clouds consumed statistically significantly less power when the FCFS mechanism (median = 506 kWh) was used than the FT mechanism (median = 1312 kWh), $Z = -11.64, \mathrm{p} < .001$, at the 95% level of confidence.

Table 6-16: Power Consumption Metric for FT Mechanism

Cloud Type	Mechanism	Mean (kWh)	Median (kWh)	Var	Std Dev	SE	K-S Test
Private Cloud	FCFS	533	538	867.4	26.45	2.2	$P < .001$
	FT	1112	1108	1185.42	34.43	2.57	$P = .011$
Public Cloud	FCFS	507	506	131.85	11.48	.86	$P = .2$
	FT	1309	1312	1124.04	33.53	2.5	$P = .006$

Therefore, it can be concluded that the power consumption of the PMs of both private and public Desktop Clouds increased statistically significantly when the FT VM allocation mechanism was employed, compared to the FCFS mechanism.

6.2.1.3 Availability

This subsection studies the impact of replication technique used by the FT mechanism on availability of nodes in Desktop Clouds by comparing the FT mechanism with the Greedy mechanism in private (NotreDame) and public (SETI@home) Desktop Clouds. For the private Cloud, the results of the availability metric were not statistically significantly normal: $\mathrm{p} < .05$. Therefore, the Wilcoxon test was conducted to compare the median of availability output. Nodes in the NotreDame Desktop Cloud were statistically significantly more available when the Greedy mechanism (median = 86.22%) was used than when the FT mechanism was used (median = 74.68%): $Z = -11.64, \mathrm{p} < .001$, at the 95% level of confidence. For the SETI@home Desktop Cloud, the T-test was applied to compare the means of the Greedy and the FT mechanisms, because the results followed a normal distribution: $\mathrm{p} > .05$.The test showed that availability of nodes when the Greedy mechanism was employed (mean = 92.59%, SE = .02) was statistically significantly better than when the FT mechanism was employed (mean = 85.28%, SE = .04) was used:$t(179) = 194.09, \mathrm{p} < .001$, at the 95% level of confidence. Table 6-17 reports the mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) availability metric results for the Greedy and the FT mechanisms for private and public Clouds.

Table 6-17: Availability Metric for FT Mechanism

Cloud Type	Mechanism	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
Private Cloud	Greedy	86.22	86.23	3.09	1.76	.13	P < .001
	FT	74.35	74.68	10.51	3.24	.24	P < .001
Public Cloud	Greedy	92.59	92.6	.1	.31	.02	P = .2
	FT	85.28	85.33	.44	.67	.05	P = .2

6.2.2 Discussion

This experiment demonstrated that the FT mechanism improved the throughput of private and public Desktop Clouds. The FT mechanism increased the throughput of NotreDame Desktop Cloud by 6% above that achieved using the Greedy mechanism, as the average outcome of throughput was 99.89% when the FT mechanism was used, and increased throughput (mean = 99.88%) of the SETI@home Desktop Cloud by 16% above that achieved with the Random VM mechanism. The FT mechanism uses a replication technique that ensures that there are at least two copies of the same VM running on different PMs. So, if a PM fails there are copies of the hosted VMs of this PM running on another PM(s). However, there is a chance that both the primary VM and the replica of a VM fail at the same time if their hosted PMs fail simultaneously. The probability is influenced by two factors: the number of PMs running and the number of node failure events. However, the probability is slight, since the proportion of lost tasks in private and public Desktop Clouds is .11% and .12% respectively, as the experiment showed.

The experiment also studied the effectiveness of the FT mechanism in terms of power consumption. It doubled the number of running VMs, thus more power was consumed. Nodes in the NotreDame Desktop under the FT mechanism (mean = 112 kWh) consumed, on average, more than that under the FCFS mechanism (mean = 533 kWh). Similarly, the amount of power consumed by SETI@home Desktop Clouds when the FT mechanism was used (mean = 1309 kWh) was, on average, 120% that under the FCFS mechanism (507 kWh). The explanation for the FT mechanism consuming large amounts of power is that the FT mechanism always doubles the number of VMs running and ensures that the primary VM and its replica are never allocated to the same PM which increases the power consumption even further, as described in section 4.3.4. Figure 6-3 and Figure 6-4 show the range of power consumption by nodes when the FCFS and FT mechanisms were employed in the NotreDame and SETI@home Desktop Clouds, respectively. They show that under the FT mechanism consumption was 1200–1400 kWh in both Desktop Clouds, while the range under the FCFS mechanism was only 450–600 kWh. This demonstrates the need to implement a policy within the FT mechanism to reduce power consumption by nodes in Desktop Clouds.

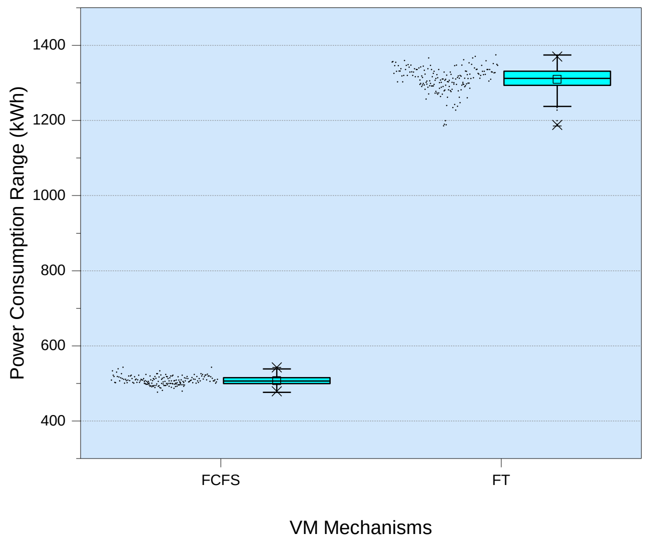

Figure 6-3: Power Consumption Range for FCFS and FT in NotreDame System

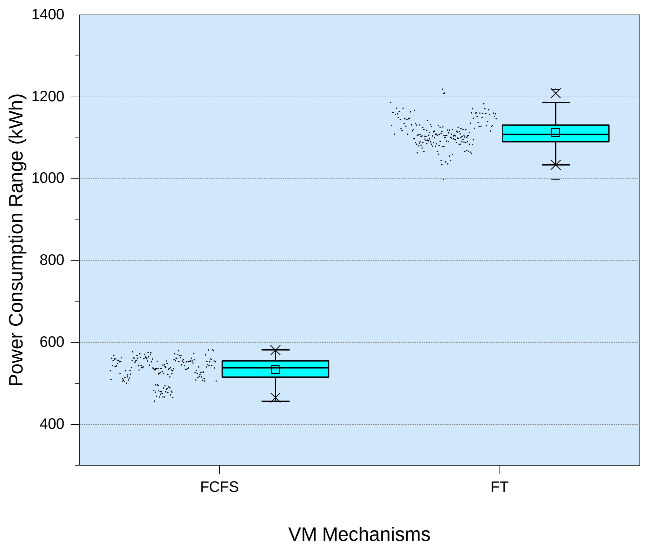

Figure 6-4: Power Consumption Range for FCFS and FT in SETI@home System

The availability of nodes in Desktop Clouds was also assessed in this experiment. Availability was negatively affected by the FT mechanism. This is due to the replication technique, which doubled the number of VMs running. The availability decreased by 12% in the NotreDame Desktop Clouds' nodes when the FT mechanism (mean = 74.35%) was used, compared to using the Greedy mechanism (mean = 86.22%). Similarly, the availability decreased by 7%. It is worth mentioning that a decrease of availability is the price that has to be paid to improve the throughput of Desktop Cloud systems: there is no way to avoid it.

The experiment demonstrated that the FT mechanism improved the throughput of both private and public Desktop Cloud systems statistically significantly more than traditional VM allocation mechanisms. However, the replication technique employed meant the improvement comes at a price, the consumption of considerable power. Question *Q2.1* is answered by saying that the FT mechanism led to greater energy consumption by PMs in Desktop Cloud systems than other mechanisms, therefore it is recommended that a technique is adopted to minimise this effect. In addition, the experiment showed that availability was affected by using the replication technique, answering Question *Q2.2*.

6.3 Experiment III: Utilisation-Based Migration Policy

The third experiment aimed to decrease the amount of power consumed by nodes when the FT mechanism was employed. The previous experiment showed that the nodes of Desktop Clouds consumed a considerable amount of power when the FT mechanism was used, because it replicated the running VMs. This experiment tried to find a utilisation threshold to enable a reduction of the power consumption with minimal loss of throughput in the

Desktop Cloud system. Section 4.4 explained that UBMP can help reduce by improving the utilisation level of these nodes. Hypothesis H3 was investigated in this experiment:

> *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with an accepted decrease in the throughput outcome*

There are several questions relating to the aforementioned hypothesis that this experiment aimed to answer:

> *Q3.1 –What is the most suitable node utilisation threshold to reduce consumption of power with minimal impact on throughput?*
> *Q3.2 –What is the impact of online VM migration policy on the throughput of a Cloud system?*

In order to find the most suitable utilisation threshold to reduce power consumption with minimal reduction of throughput, an online migration policy was implemented. The following section describes the methodology of this experiment, while the next analyses the results. Third, there is a discussion of the experiment findings.

The experiment sets a utilisation threshold at eight values, as listed in Table 6-18, and ran each utilisation threshold separately. The maximum utilisation is set to 70%, because this level is mentioned as the optimum utilisation threshold for power consumption by [106]. In addition, setting the utilisation level above 70% can cause extra overheads due to the number of VM migrations required. The PMs running were scanned periodically. If a PM was under-utilised at the set threshold value, then all hosted VMs were migrated to another PM. The results of throughput and power consumption metrics were analysed in order to find which utilisation threshold gave the best results.

Table 6-18: Utilisation Threshold Values

Utilisation Threshold
0%
10%
20%
30%
40%
50%
60%
70%

6.3.1 Results

The results of throughput and power consumption metrics of NotreDame Desktop Cloud are analysed in the first subsection in order to find the best throughput-power consumption trade. The second subsection analyses the results of SETI@home Desktop Cloud. (5). Power consumption was calculated as in equation (6).

6.3.1.1 NotreDame Desktop Cloud

According to the K-S normality test, the results of the throughput metric shown in Table 6-19 did not follow a normal distribution: P-value < .05. Therefore, Friedman's ANOVA was selected to test if there was a statistically significant difference between results. Friedman's ANOVA test showed that throughput varied significantly from one mechanism to another: $X_F^2(7) = 538.37, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-19. However, the results demonstrated that changing the utilisation level from 10% to 70% had only a slight impact on throughput.

Table 6-19: Throughput Metric results for NotreDame Cloud for FT Mechanism with Different Utilisation

Utilisation	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
0%	99.89	99.99	.09	.29	.02	P < .001
10%	99.91	99.99	.06	.23	.01	P < .001
20%	99.86	99.99	.12	.35	.03	P < .001
30%	99.8	99.99	.29	.54	.4	P < .001
40%	99.68	99.99	.6	.77	.6	P < .001
50%	99.34	99.71	1.65	1.28	.1	P < .001
60%	99.25	99.52	1.69	1.3	.1	P < .001
70%	99.28	99.57	1.26	1.12	.08	P < .001

According to the K-S normality test, the results of the power consumption metric shown in Table 6-20did not follow a normal distribution, because the results of both 0% and 30% were that P <. 05. Therefore, Friedman's ANOVA was selected to test the null hypothesis, which is that there was no statistical significant difference between the results. The null hypothesis was rejected, because $X_F^2(7) = 692.78, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-20. The mean and the median of power consumption when the utilisation threshold was set to 40% consumed statistically significantly the least power. Therefore, seven pairwise post-hoc tests were conducted to confirm whether there was a statistically significant difference between the results when the utilisation level was 40%, compared with other levels. The Wilcoxon test was applied in the seven post-hoc tests and showed indeed that there were statistically significant differences between the results of the power consumption metric when the utilisation threshold was 40% and when the threshold was set to 0%, 10%, 20%, 30%, 40%, 50%, 60% and 70%, when P-value < .0018. The level of significance was adjusted to 0.0018 using the Bonferroni correction, because 28 post-hoc tests were needed to confirm that there was a statistically significant difference between each utilisation threshold in the experiment. That made the critical value P = .05/28 = .0018. Therefore, it can be concluded that the power consumption of nodes when the utilisation threshold is set to 40% has a minimal effect on throughput in the NotreDame Desktop Cloud.

Table 6-20: Power Consumption Results for NotreDame Cloud for FT Mechanism with Different Utilisation

Utilisation	Mean (kWh)	Median (kWh)	Var	Std Dev	SE	K-S Test
0%	1112	1108	1185.42	34.43	2.57	$P = .011$
10%	1109	1106	1210	34.78	2.59	P = .02
20%	1103	1102	1342	36.63	2.73	P = .2
30%	1049	1060	2343	48.41	3.61	P < .001
40%	989	989	1933	43.96	3.28	P = .2

50%	1117	1117	6791	82.41	6.14	P = .2
60%	1170	1168	5948	77.12	5.75	P = .2
70%	1170	1166	5996	77.44	5.77	P = .2

6.3.1.2 SETI@home Desktop Cloud

The results shown in Table 6-21 for the throughput metric for the SETI@home Desktop Cloud were not normally distributed, according to the K-S normality test: P <.05. So, Friedman's ANOVA was applied to test the null hypothesis, which was that there was no statistically significant difference between throughput outcomes at various utilisation thresholds. The test rejected the null hypothesis, because $X_F^2(7) = 1158.22, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-21. The median of throughput values under the mechanism when utilisation was set to 0%, 10%, 20%, 30% and 40% was within the accepted range, at 99.2% or above. However, this was not the case when the utilisation thresholds were at 50%, 60% and 70%, as the median of throughput values decreased to less than 96%.

Table 6-21: Throughput Metric Results for SETI@home Cloud for FT Mechanism with Different Utilisation

Utilisation	Mean (%)	Median (%)	Var	Std Dev	SE	K-S Test
0%	99.88	99.99	.1	.32	.02	P < .001
10%	99.99	99.99	.01	.02	.01	P < .001
20%	99.9	99.99	.01	.01	.01	P < .001
30%	99.88	99.88	.05	.04	.02	P < .001
40%	99.24	99.33	.32	.02	.04	P < .001
50%	95.55	95.83	3.55	.04	.14	P = .02
60%	91.1.7	92.02	7.26	.2	.2	P = .2
70%	91.52	91.52	8.03	.21	.21	P = .2

The results of the power consumption metric for SETI@home Desktop Cloud were not normally distributed, according to the K-S normality test: P <.05. So, Friedman's ANOVA was applied to test whether there was a statistically significant difference between nodes' power consumption when different utilisation thresholds were applied. The test rejected the null hypothesis, because$X_F^2(7) = 1158.22, P < .001$, at the 95% level of confidence. Mean, median, variance (Var), standard deviation (Std Dev) and standard error (SE) are reported in Table 6-22. As the table shows, nodes consumed less power when utilisation was set to 30% and 40%, because they consumed similar to the median of 1003 and 1033 kWh when utilisation was set at 30% and 40%, respectively. Several post-hoc analyses with Wilcoxon signed-rank tests were conducted with a Bonferroni correction applied, giving a significance level adjusted to P = .05/28 = .0018. Although the post-hoc tests showed a statistically significant difference between the results when utilisation was set to 30% and 40%, it showed no statistically significant difference between the results (i.e. when the utilisation level = 30% versus utilisation level = 40%). Therefore, it can be concluded that nodes consumed the least power in SETI@home when the utilisation level was set to 30% or to 40%.

Table 6-22: Power Consumption Results for SETI@home Cloud for FT Mechanism with Different Utilisation

Utilisation	Mean (kWh)	Median (kWh)	Var	Std Dev	SE	K-S Test
0%	1309	1312	1124	33.53	2.5	P = .006
10%	1255	1259	955	30.9	2.3	P = .001
20%	1209	1210	577	24.01	1.79	P = .2
30%	1005	1003	1096	33.11	2.47	P = .2
40%	1036	1033	1516	38.94	2.9	P = .2
50%	1347	1344	5196	72.09	5.37	P = .2
60%	1509	1512	1823	42.69	3.18	P = .2
70%	1509	1513	1488	38.57	2.88	P = .064

6.3.2 Discussion

This experiment tried to find a utilisation threshold to help to reduce the power consumed by nodes. It was shown in section 4.4 that improving utilisation can reduce power consumption. However, the main problem with node failure is that allocating many VMs to the same PM may affect throughput if the PM fails. Therefore, this experiment was conducted in two types of Desktop Clouds, private and public, in order to find a utilisation threshold that reduced power consumption yet maintained an acceptable level of throughput.

In the NotreDame Desktop Cloud system, changing the threshold values from 0% to 70% caused a minor degradation of the throughput metric of only .5% maximum. In order for tasks to be completely lost, reducing throughput, the host of the primary VM and the host of its replica have both to fail at the same time, which did not happen in the case of NotreDame Desktop Cloud for two reasons. The first reason is that the level of node failure was not high, at a rate of only 6% of nodes per hour. The second reason is that even if several VMs were allocated to the same PM, this did not necessarily mean that their replicas were allocated to a single PM; they would be allocated to different PMs. This would make the chances very low that two PMs, one hosting a primary VM and the other the replica VM, would fail at the same time. Therefore, the selection of the utilisation threshold depended only on the power consumption metric. Table 6-20 shows the power consumption of NotreDame nodes. Nodes consumed the least power when the utilisation threshold was set to 40%.

It may be asked why the nodes consumed more power when utilisation was set to 50%, 60% and 70%. This was because of what may be termed ‘over migration’, meaning migrating VMs from nodes with a utilisation level below the threshold value to another node(s), without actually increasing the utilisation threshold. For example, suppose the utilisation threshold is set to 70%. In the case of a node with 60% utilisation, all hosted VMs have to migrate, yet there is the possibility that no nodes can accommodate them because they are almost fully occupied (the threshold is set to 70%). Therefore, these VMs might go to nodes with lower utilisation thresholds, or might even be allocated to nodes hosting no VM at all.

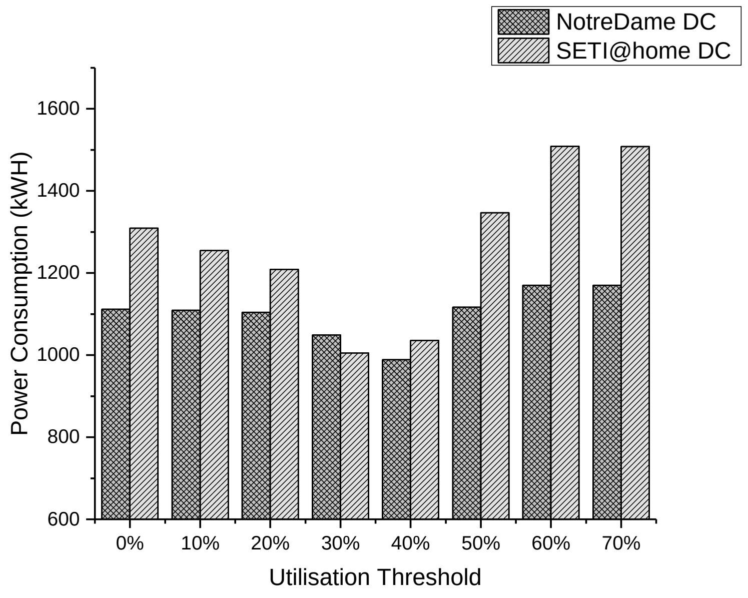

Figure 6-5: Power Consumption According to Utilisation Thresholds

In the SETI@home Desktop Cloud it is a slightly different case, because throughput varied according to the utilisation threshold, as Table 6-21 shows. Utilisation thresholds with values at 50%, 60% and 70% were excluded because they have less than acceptable levels of throughput. Nodes in the SETI@home Desktop Cloud consumed the least power when the utilisation power was set at 30% or 40%, and this can be statistically concluded, as shown in section 6.3.1.2. Nevertheless, the difference between these two values in terms of power consumption is quite small, at only 3% of power. Therefore, it may be said that the utilisation threshold is 40%, because it is the same figure as in the case of the NotreDame Desktop Cloud, as depicted in Figure 6-5.It is worth mentioning that the power saved in the case of SETI@home was 20%, on average, while it was 10% in the NotreDame Desktop Cloud. The reason behind the double saving in the SETI@home Desktop Cloud is the double number of nodes of SETI@home (872 nodes) compared to the NotreDame Desktop Cloud (472 nodes), as stated in section 5.3.2. Therefore, it may be said that a utilisation threshold is beneficial as the number of Desktop Clouds grows ever greater.

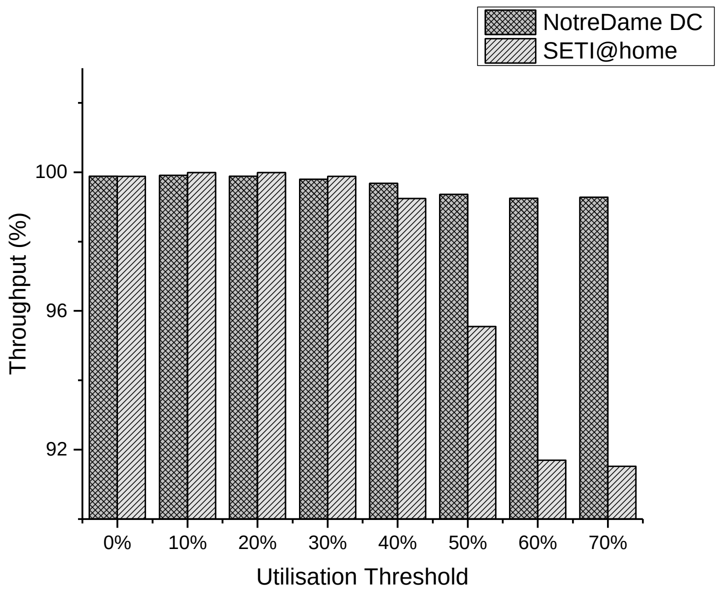

Figure 6-6: Throughput According to Utilisation Thresholds

In addition, Figure 6-7 and Figure 6-8 demonstrate the range of power consumption when the FT mechanism was employed with different utilisation threshold levels for the NotreDame and SETI@home Desktop Clouds. The figures are box plots of the power consumed for nodes, from the experiment. The small dots represent the daily average power consumption for a Desktop Cloud node, while the two ends of each box plot represent the maximum and minimum values. Figure 6-8 shows that having utilisation thresholds at 30% and 40% clearly outperforms other thresholds by far; even the maximum power consumption values were below the minimum power consumption achieved using other utilisation thresholds. Such notation confirms the finding that a utilisation threshold at 40% can be beneficial when the number of running nodes in a Desktop Cloud system is great.

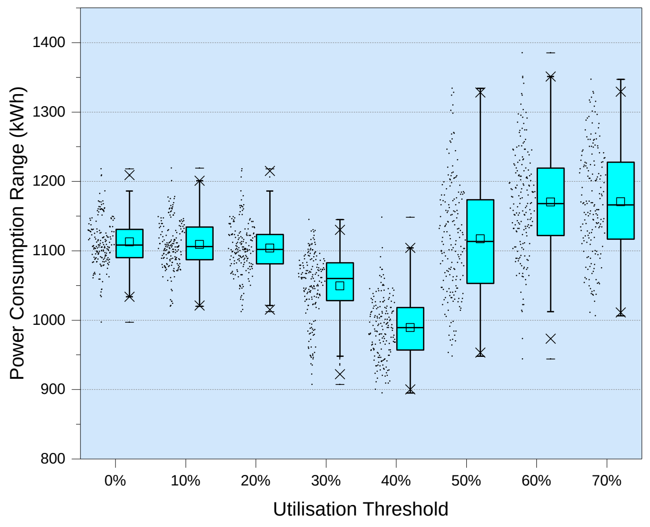

Figure 6-7: Power Consumption Range for FT Mechanism with Different Utilisation Thresholds in NotreDame System

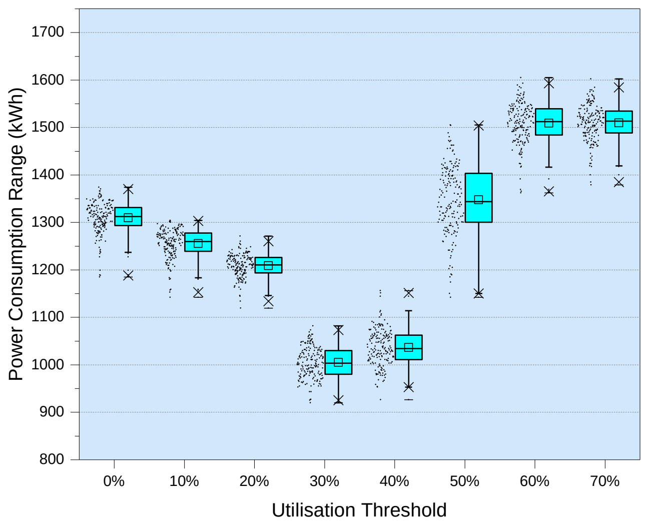

Figure 6-8: Power Consumption Range for FT Mechanism with Different Utilisation Thresholds in SETI@home System

Utilisation is used in this migration technique. If a PM has utilisation above zero but below a certain level, all VMs allocated to this PM will be migrated to another in order to improve utilisation. Question *3.1* is answered by identifying a critical utilisation level of 40%. At this level there is optimal trade-off between throughput and power consumption. The experiment found that setting this threshold in UBMP statistically significantly reduced power consumption by about 20%, with the penalty of a maximum drop in throughput of 2%, answering Question *3.2*.

6.4 Chapter Summary

This chapter presented the evaluation part of this research, including the methodology of each experiment, the obtained results and analysis. Three experiments were conducted. The first experiment aimed at evaluating the impact of node failure on Desktop Cloud systems using the metrics of throughput, power consumption and availability. Four VM allocation mechanisms – the FCFS, Greedy, RoundRobin and Random, as retrieved from the literature – were implemented in this experiment to find which gave the best results in each evaluation metric separately, once in a private Desktop Cloud (using NotreDame dataset) and once in a public Desktop Cloud (using the SETI@home dataset). The results showed that the Greedy mechanism gave the best results for throughput, but for of availability the private Desktop Cloud gave the best results, in both private and public Desktop Clouds. The Random mechanism gave the best results for throughput in a public Desktop Cloud. The FCFS mechanism consumed the least power in both private and public Desktop Clouds. The experiment emphasised that throughput in a Desktop Cloud system may be reduced due to node failure events.

The second experiment studied the proposed mechanism, the FT mechanism, which aimed to improve throughput in a Desktop Cloud system. The first experiment showed that node failure can reduce throughput by up to 20% of submitted tasks. This experiment compared the outcome of Desktop Cloud systems when the FT mechanism was employed against the best mechanisms obtained from the first experiment. According to the results, the FT mechanism managed to improve throughput in both private and public Desktop Clouds. However, such improvements came at a price, represented by the increase of power consumed by Desktop Clouds' nodes. Such an increase shows the need to implement a solution to reduce the power consumed by nodes.

The third experiment tried to find a solution to the power consumption issue in the FT mechanism by setting a utilisation threshold to trigger migration of VMs, with the aim of improving resource utilisation. The results showed that power may be saved when the utilisation level was set to 40%. In a public Desktop Cloud, on average 20% was saved as a result of implementing the OUMP. The three experiments confirmed the three hypotheses:

> H1: *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds*
>
> H2: *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud system*
>
> H3: *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with a decrease of 2% in the throughput outcome*

The next chapter discusses the findings of this research to review them alongside the challenges of Desktop Clouds introduced in section 2.3.4. The research contributions are described, with the aim of showing how they match and overcome the research gaps in the field of Desktop Clouds.

Chapter 7: Discussion of Research Findings

Chapter 2 introduced the Desktop Cloud as a new type of system that employs idle computer nodes as PMs, discussing several the research challenges brought about by Desktop Cloud computing that need to be tackled before it can represent a viable alternative to Traditional Cloud computing. Node failure was identified as something to be solved and Chapter 3 reviewed various VM allocation mechanisms, policies and solutions proposed in the field of Cloud computing to show that none can be employed in Desktop Clouds to solve this issue. Chapter 4 proposed a novel fault-tolerant mechanism to reduce the effect of node failure on the outcomes of a Desktop Cloud system.

Chapter 5 presented the methodology of this research to test the proposed VM mechanism in order to reflect its effectiveness in three metrics: throughput; power consumption; and availability. Chapter 6 gave the results of experiments based on the methodology of Chapter 5 to test the hypotheses of this research, and explained how each experiment was conducted.

This chapter discusses the findings of this research and positions them alongside related findings in the literature. The discussion illustrates that the contribution of this research is to fill gaps in the research into Desktop Cloud computing and it starts by discussing the impact of node failure in Desktop Clouds, with an analysis of the number of nodes that can fail at any given time in either private or public Desktop Cloud systems. Next comes a discussion of the difference between Desktop and Traditional Cloud systems using the DesktopCloudSim simulation tool. The proposed evaluation metrics are discussed in terms of their ability to assess VM allocation mechanisms. The chapter next discusses the capacity of the designed FT mechanism to tolerate node failure in Desktop Cloud systems. Further, a discussion proposes how the UBMP can be employed by the FT mechanism to reduce power consumption. Lastly, the chapter discusses the limitations of the research.

7.1 Impact of Node Failure

Several studies into Desktop Cloud computing point out that node failure can have a negative impact upon the outcomes of a Desktop Cloud system [11], [60], [180], but none gives sufficient evidence to support this claim, thus this research first investigated through empirical study the impact of node failure using real-world failure traces. This research employed a simulation tool to evaluate the outcome of Desktop Cloud systems using throughput, power consumption and availability metrics.

7.1.1 Failure Analysis

In this research, a study of node failures in PMs at Desktop Grids systems was carried out: (1) to compare failures events in private and public Desktop Clouds; (2) give figures to explain throughput variations in different VM allocation mechanisms; and (3) analyse the failure to help researchers to design a prediction mechanism to migrate VM instances from PMs that are predicted to fail before they fail. The studies concluded that node failure in public Desktop Grid systems is double that of private Desktop Grid systems, as section 5.3.2 demonstrated. However, related failure studies focus on the time between failures in PMs in Desktop Grid systems. The findings of Kondo et al., [22] were that, on average, the failure ratio was similar to that in this study for private Desktop Grid systems; however, their Desktop Grid system differed from that in this research, although considered a private system. They conducted an experiment to study how long a host would remain available before it failed in the Diego Supercomputer Centre Desktop Grid system, examining 220 hosts over a period of 28 days. The study showed that, on average, 5% of hosts failed. However, the study did not focus on the number of nodes failing but the duration that a node remained live.

The ratio of node failure in public Desktop Cloud systems can reach double that in private Desktop Cloud systems. There are several reasons. First, participants are under no obligation to remain working in a public system, unlike in private systems where regulations may be imposed to ensure that nodes remain connected. For example, a university can require every computer node belonging to it to join the system, Desktop Cloud or Grid, when it becomes idle. Another reason is that PMs in public systems are connected via the Internet, which increases the probability of connectivity failure, while PMs in private systems usually have a LAN connection. Furthermore, the number of nodes joining a public Desktop Cloud system can be high, which means there are more PMs to fail.

The failure study conducted by this study makes a twofold contribution to the field of Desktop Cloud research. First, the study focuses on the number of nodes failing during a given time period (in this case, hourly). Having figures of failing nodes available can yield better knowledge about how to design a VM allocation mechanism. For example, section 5.3.5 showed that the number of VM replications required is identical in both private and public Desktop Clouds and can rise if the number of failing nodes is higher, as section 6.2 revealed. The second advantage is that having statistical failure trends helps to predict when a PM is going to fail. This can be employed in the migration policy UBMP in order to let a VM instance migrate from a PM that is predicted to fail soon to another PM. Such prediction studies have been proposed in the area of Desktop Grids such as [172], [181] and [173], yet not in the area of Desktop Clouds.

7.1.2 DesktopCloudSim

Simulation was the methodology adopted by this study to evaluate the solution proposed for a VM mechanism tolerant to node failure. Simulation was also used to test the hypotheses, as demonstrated in section 5.3. Therefore, this research has extended use of the well-known Cloud simulation tool, CloudSim. Section 5.45.3 showed that it assumes that PMs in a simulated Cloud system are reliable, with no node failure, thus it does not simulate node failure. This study developed DesktopCloudSim as an extension tool of CloudSim in order to simulate node failure and thus simulate Desktop Cloud systems.

Previous sections discussed node failure in private and public Desktop Cloud systems. This section discusses its effect on the outcomes of a Desktop Cloud system, specifically the throughput metric. Therefore, a baseline experiment was conducted to assess the impact of node failure by using DesktopCloudSim to compare outcomes of Traditional Cloud and Desktop Cloud systems. It illustrated the impact of node failure on throughput outcomes and showed that DesktopCloudSim is capable of simulating node failure. Section 5.4 describes the experiments in order to answer the following question:

> *What is the difference between Desktop Clouds and Traditional Clouds in terms of throughput, power consumption and availability?*

The baseline experiment tested three VM allocation mechanisms (FCFS, Greedy and RoundRobin) by implementing each once in simulated Desktop Cloud systems and once in simulated Traditional Cloud systems in order to answer the previous question. The experiment concluded that the main difference is in terms of throughput, which was affected badly by node failure regardless of the mechanism employed. This comparison of Desktop and Traditional Cloud systems using empirical data is considered the first step in this field, so there are no other studies with which to compare the findings. Nevertheless, it can be said that node failure can cause a decrease in the throughput metric of a Desktop Cloud system by reducing the number of successfully processed tasks, as section 5.4.4 demonstrated. Moreover, the throughput of public Desktop Cloud is less than that of a private Desktop Cloud. The node failure ratio plays a key role in the variation of throughput outcomes between private and public Desktop Cloud systems; section 5.3.2 showed that the average node failure ratio in a public Desktop Cloud is at least double that in a private Desktop Cloud system.

7.1.3 Evaluation Metrics

Section 4.5 proposed throughput, power consumption and availability metrics to evaluate the outcome of a VM allocation mechanism in Desktop Cloud systems. Section 6.1 showed the results of an experiment conducted to test the Hypothesis H1:

The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds.

Throughput means the number of successfully completed tasks. Throughput is an important metric in designing VM allocation mechanisms in Desktop Clouds, unlike Traditional Clouds; because of node failure issue. The authors in [182] consider it an important factor for Desktop Grids for this reason. A finding of this study confirmed that node failure affects throughput when using a VM allocation mechanism. The experiment compared four VM allocation mechanisms: FCFS, Greedy, and RoundRobin, the usual mechanisms in open-source Cloud management platforms, with a fourth mechanism, Random, which is a naïve mechanism that randomly selects and assigns VM instances to PMs. The experiment showed that, regardless of which of the four VM allocation mechanisms was employed, throughput was affected by node failure because when a failure occurs in a PM, all hosted VM instances are destroyed. Therefore, the tested VM allocation mechanisms cannot be used to reduce the effect of node failures since they do not employ a replication technique. It is clear that there is a real need to design a mechanism able to mitigate the effects of node failure during runtime by employing replication.

The power consumption metric concerns the amount of energy consumed by PMs in a Desktop Cloud system. It is considered an important factor by which to evaluate the behaviour of a VM allocation mechanism in Traditional Cloud systems in order to reduce maintenance costs, as discussed in section 3.2.2, so it is crucial to consider power consumption when it comes to assessing VM mechanisms. Experiment I was conducted to compare the usual VM allocation mechanisms. It was seen that FCFS consumed the least power because it allocates VM instances on a BF basis; the literature shows that this solution improves resource utilisation, reducing power consumption according to [183]. Therefore, it is recommended that this is employed in the FT solution of this study in order to improve resource utilisation.

The availability metric measures the amount of computing power of a Desktop Cloud system for new requests from users. It is important to consider availability in designing a VM allocation mechanism to assess the consequences of employing a replication technique, because this increases the number of running VM instances so there is less free computing power of PMs available to serve new requests to accommodate VM instances. So, the availability metric is used in this study to assess the impact of employing a replication technique.

7.2 FT mechanism

The FT mechanism developed in this study is considered to be the first attempt to tackle node failure in Desktop Clouds, and it is compared to traditional VM allocation mechanisms since there are no others available. However, these pay little attention to node failure and thus do not employ replication techniques. The authors in [184] identified the need to implement a fault tolerance policy as one of the challenges facing design middleware for Desktop Cloud systems and the findings of this research, as presented in section 2.3.4.4, confirm this finding. Chapter 4 proposed a new VM mechanism tolerant of node failure. The mechanism employed a replication technique, which is regarded as resilient of node failure. Its main strategy is that, no matter how many nodes fail during runtime of a VM instance, there is always a replica for a particular VM ready to take over.

Table 7-1: FT Mechanism vs. Related Works

Criteria / Mechanism	Throughput	Performance	Cost	Utilisation	Faulty Resources
FT Mechanism	√	X	√	√	√
Work by [106]	X	√	X	√	X
MBFD Algorithm [86]	X	√	√	√	X
RoundRobin Policy [118]	X	X	√	X	X
FF Policy [185]	X	√	X	√	X

The FT mechanism is effective in improving throughput, utilisation of PMs and reducing running costs in the presence of node failures. Running costs is handled by the FT mechanism by reducing power consumption. However, the mechanism does not pay attention to performance of running service. A summary of comparison between the FT mechanism and related works is listed in Table 7-1 using these criteria. The related works were presented in chapter 3.

Throughput in the table is related to faulty resources. Therefore, it is overseen by related works, discussed in chapter 3, because high reliability of nodes is assumed. Utilisation is considered by this work in order to reduce power consumption with a goal to reduce running costs. This goal is shared with related works as discussed in section 3.2.2. However, the work presented by this research differs by the fact that it trades off between improving throughput by using replication technique and reducing power by improving utilisation. Although, the FT mechanism improves throughput in the presence of node failures and reduces power consumption, the mechanism, however, pays no attention to the performance criteria. Performance can involve response time, SLA violations .. etc, as it is discussed in section 3.2.3. The FT does not consider performance because the main focus in this research is to improve throughput in faulty resources. In addition, it can be argued that in faulty systems, such as Desktop Grids and Desktop Clouds, performance is ignored in return for improving throughput according to [186]. Related works, listed in the table, focus on the performance because they are developed for Traditional Cloud systems which assumed high reliability of nodes. Nevertheless, the FT mechanism is considered a step to improve throughput that can be followed by extensions to improve performance as future works.

7.2.1 Static FT Mechanism

The first phase of the mechanism is the static mechanism, which aims to improve only throughput. The mechanism of PM selection for both primary and replica VM instances plays a key role for two reasons. The first is that it can help to reduce the number of destroyed primary VM instances. The second is to reduce the power consumed PMs by improving resource utilisation. Section 4.3 explained that the FT mechanism employs a BF heuristic solution to select PMs with highest utilisation level to accommodate VM replicas. According to [183], this heuristic can improve resource utilisation by stacking VM instances to the same PM(s), leading to a reduction in power consumption.

The LF heuristic solution is employed to select PMs to host primary VMs in the FT mechanism in order to distribute primary VM instances to PMs. The aim of using this heuristic method is to improve the resilience of the mechanism against node failure, because fewer primary VM instances being destroyed leads to a smaller effect on throughput. Therefore, the PM selection phase in the FT mechanism aims for a trade-off between throughput, using the LF solution, and power consumption, using the BF solution. Throughput increased, according to section 6.2. The second hypothesis of this research. H2 was tested in the section:

> *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud System*

Experiment I was conducted to examine this hypothesis, which demonstrated that the FT mechanism statistically significantly improved throughput outcome in Desktop Cloud systems in comparison to other traditional VM allocation mechanisms. It can be seen that the FT mechanism reduced the effect of node failure in both private Desktop Clouds (with a low level of node failure) and public Desktop Clouds (with higher level of node failure). The FT mechanism ensures that there is always a replica for a running VM instance. If a node is reported as failing, then the mechanism scans the VM instances to check and replicate if one is found destroyed. The throughput outcome of the FT mechanism was kept above 99% in both private and public systems. Complete failure status is reached when there is loss of throughput in the mechanism and means that both the PM hosting a primary VM instance and its replica fail at the same time. From the experiment, it can be seen that the probability of complete failure status is about 0.11% when the average failure of nodes is about 6.26% (as in private Desktop Cloud systems) and about 0.12% when the average failure of nodes is about 13.67% (as in public Desktop Cloud systems).

7.2.2 UBMP

The second experiment demonstrated that the static FT mechanism improved the throughput of both private and public Desktop Cloud systems. However, this mechanism causes PMs to consume energy, because it pays little attention to resource utilisation. This raises the question in this context whether throughput outcome is affected if primary VM instances stack together. The FT mechanism was extended to include the UBMP, implemented as a dynamic method to migrate VM instances in order to improve resources utilisation during runtime. The third experiment was conducted to investigate the hypothesis H3:

> *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with an accepted decrease in the throughput outcome.*

The hypothesis employs "accepted" term to describe the decrease of throughput outcome because the maximum decrease can reach up to 2%, in worst case, of all submitted tasks. Such figure is considered accepted in Desktop Cloud systems in return for the power saving of about 20%. It was pointed out by [86] that about 3% of SLA violations is accepted in Traditional Cloud in return for a power saving of about 30%. Therefore, the same figure can be applied to Desktop Cloud systems by stating that a downgrade of about 2% of throughput outcome in return for 20% of power consumed by nodes seemed to be accepted.

The experiment tried several utilisation thresholds from 0 to 70% in order to find the optimum point for the trade-off between throughput and power consumption for both private and public Desktop Cloud systems. It can be seen from the experiment that increasing utilisation can result in a decrease in throughput for a Desktop Cloud system, as demonstrated in section 6.3. It may also be claimed that increasing resource utilisation increases the risk of total failure; as there are more VM instances stacked to fewer PMs, failure of a single PM can cause multiple VM instances to fail at the same time.

When the failure ratio is relatively low (in the case of a private Desktop Cloud), increasing utilisation to 70% has a minor impact (less than 1%) on throughput, as shown by the third experiment. However, there are other dimensions that affect throughput, namely the number of PMs and the number of VM instances running. The availability metric can be used in this context to solve the issue; it can be seen that when the level of failure in a Desktop Cloud is quite low (6% or less), increasing utilisation will not decrease throughput provided availability is above 86%, as demonstrated in section 6.3.

The migration of VM instances, as used in the UBMP, has two drawbacks. The first is performance degradation during migration, as discussed in terms of Traditional Cloud computing by authors such as [86], which can cause violations of SLA between CSPs and users. There is still the same issue of performance degradation in Desktop Cloud systems, but its impact can be ignored because users of such systems expect a poorer quality of service in return for the low cost of exploiting services from Desktop Cloud systems. The second drawback of using online migration is the data transfer impact upon network because of moving VM instances around PMs. However, the authors in [78] adopt an approach to VM transfer mechanism that can reduce the impact of migration upon the network. Their work is useful in two ways: it helps to reduce the cost of keeping primary VMs and their replicas synchronised, and to minimise the data required to transfer to PMs when failure occurs.

The authors in [92] state that allocating many VMs to the same PM can cause performance degradation of these VM instances. However, the FT mechanism is not concerned with this possible issue, because (1) performance degradation of VM instances is tolerable from the point of view of the end users; and (2) some researchers (according to [93] argue that the performance degradation from consolidating VMs to the same PM is quite limited and thus can be ignored.

7.2.3 Running Cost

The cost of running an entire Desktop Cloud system needs further attention in order to study if it is worth choosing over Traditional Cloud systems. This research addresses power consumption as a metric for the outcome of the proposed mechanism. Indeed, power consumption contributes large amount of running costs in both Desktop and Traditional Cloud systems [2]. However, there are several factors that can affect the cost of running both

Desktop Grid and Desktop Cloud systems [187]. First of all, the cost of purchasing hardware to run and monitor systems and consequently the cost of maintaining these machines. In addition, the salary of staff who install and program of software for Desktop Cloud systems can be quite high. Finally, network bandwidth contribute as well to the overall cost of running Desktop Cloud systems.

Therefore, it can be said that although this research tackles the problem of running Desktop Cloud system by reducing power consumed by nodes within a system. There is still a need for further investigation and comparison study between Traditional and Desktop Cloud in order to demonstrate the cost effectiveness of Desktop Cloud systems.

7.2.4 Design Challenges

Section 4.1 presented several challenges that need to be considered when a VM allocation mechanism is designed. The challenges are discussed in this section to demonstrate that the FT mechanism has solved them.

Challenge 1. Replication of VM instances with the aim of reducing the impact of node failure on throughput.

The FT mechanism replicated VM instances. It improved throughput in private and public Desktop Cloud systems statistically significantly better than the FCFS, Greedy, RoundRobin and Random VM allocation mechanisms.

Challenge 2. Selection of a PM to host a replicated VM.

The FT mechanism adopted a heuristic solution to select a PM for VM replicas. The adopted mechanism aimed to improve resource utilisation in order to save power.

Challenge 3. Response to a failing PM.

When a failure is detected, the FT mechanism responds by allocating VM copies of all VM instances that were running on that failed PM in turn. When a VM instance is found to have been destroyed, the mechanism identifies its copy. If the VM copy is a primary, the mechanism creates a replica and allocates it to a PM. If it is a replica, the mechanism makes it a primary then creates a new replica and selects a PM for it.

Challenge 4. Implementation of a migration policy of VM instances during runtime to reduce the impact of the replication technique.

The UBMP was developed to overcome this challenge. This research has demonstrated it is able to reduce power consumption significantly statistically more than the static FT mechanism.

7.3 Limitations

This section presents several limitations of this research, classified into three subsections: experimental assumptions concerned with shortages of experiments; limitations of simulation discussing the possibility of testing the proposed work in a real-world system; and the quality of service overseen by this research.

7.3.1 Experiment Assumptions

Experiments were run through simulation, by augmenting failure traces to a simulated Desktop Cloud system. The failure traces are real-world traces collected from real systems used to simulate systems. The tasks are also real-world traces collected to simulate tasks submitted by end users to a Desktop Cloud system. However, there are some details missing from the simulation, such as the number of requested VM instances and hardware specifications of PMs in the system. Therefore, a number of requested VM instances is assumed, as explained in section 5.3.5, along with the specification of PMs, as section 5.3.3 showed. Results obtained from experiments will be more accurate if these details are gathered from real-world systems.

Furthermore, it is assumed that when a node fails, it is reported directly to the system in order to response with no delay. However, such an assumption needs further attention because the time required to discover a failing PM may lead to an increased possibility of complete failure. In addition, it is assumed throughout the experiments that improving resource utilisation leads to a reduction in power. However, this is not always the case in Desktop Cloud systems. For example, in the case of public Desktop Clouds it may be otherwise, because improving resource utilisation does not mean that resources with no utilisation will be in power saving mode, as there is no control over the owners to abide by this policy.

7.3.2 Simulation

Another limitation of this research is that the FT mechanism was tested on simulated systems and not applied to real Desktop Clouds. It may be argued that applying the proposed solution to a real system would demonstrate further issues and give further support to the proposed work [153]. This is difficult to achieve because there is no actual Desktop Cloud system that can used for this purpose, highlighted as a valid reason for using simulations by [151]. In addition, it would be time consuming to build a real Desktop Cloud system in order to apply the mechanism, quite beyond the time limits of this research. Nevertheless, it is recommended that FT mechanism is applied and tested on a real Desktop Cloud system by integrating the proposed mechanism with open-source Cloud management middleware such as Eucalyptus or OpenNebula.

7.3.3 Quality of Service

Although the proposed mechanism can tolerate node failure, the mechanism has a negative impact on the quality of service provided to users because of the migration policy and VM check-pointing. It has been shown that the migration of VM and VM check-pointing can cause performance degradation and increase the cost of data transfer [55]. Furthermore, the node selection phase for VM placement and migration increases the time when the number of PMs is large.

All of these factors can contribute to low quality of service for users of Desktop Cloud systems. Although it is stated in this study that users of such systems usually tolerate this in return for low running costs, it has not focused on assessing this quality of service. The throughput metric can help to evaluate the number of lost tasks, but the time required to complete a task can be useful in reflecting the quality of service. Other metrics to be considered in order to evaluate quality of service is bandwidth and latency. It is pointed out that these are important metrics for VM allocation mechanism from the network point of view [92].

7.4 Summary

This chapter presented a discussion of the findings of this research. It started with a discussion of analysis findings of node failures in order to explain the impact of node failure upon the outcomes of Desktop Cloud systems. The node failures were gathered from real-world systems used in DesktopCloudSim in order to simulate Desktop Cloud systems. The impact of node failure was evaluated using three metrics (throughput, power consumption and availability). In summary, node failure affects mainly the throughput outcome, regardless which VM allocation mechanism is employed, if no replication technique is used.

The FT mechanism was developed to fill this gap. The chapter discussed the improvement in throughput when the FT mechanism was used in a static way. However, the FT mechanism can cause PMs to consume more energy as a result of replicating VM instances. Therefore, the UBMP, online migration mechanism, was developed and employed to increase resource utilisation, leading to reduced power consumption by PMs in a Desktop Cloud system. The UBMP had to balance the throughput and power consumption metrics, as although increasing the resource utilisation threshold can decrease power consumption it also increases the risk of complete failure, which reduces throughput. In short, the FT mechanism is considered to be a first attempt to improve throughput in Desktop Cloud systems with some consideration being given to the reduction of power consumption.

This chapter's final section of discussed the limitations of this research. These can be summarised into: those involving the experiment inputs; those about just the simulation; and those about the quality of service.

The next chapter presents the conclusion of this research. It introduces future work that can overcome these limitations. Researchers still face challenges in making Desktop Clouds a viable solution.

Chapter 8: Conclusion and Future Work

The first section of this chapter provides a summary of research into failure-tolerant VM allocation mechanisms for Desktop Cloud systems. Its conclusion revisits the objectives of this study and presents Desktop Cloud computing as a novel type of Cloud computing and node failure as a research challenge. It states the metrics proposed to evaluate VM allocation mechanisms; the throughput improvement as a result of employing the FT mechanism developed by this research; and the ability of this study to improve this mechanism further by reducing power consumption. The second section of the chapter identifies future directions along which to pursue this research. It concludes with remarks about this research.

8.1 Research Conclusion

This section outlines the contribution of this study to advancing the field of Desktop Cloud research. This represents a new type of Cloud computing that provides services through non-dedicated resources. These can be any form of computing devices, such as PCs and laptops, to be used mainly when they become idle. The new direction attempts to combine two computing models, Cloud computing and Volunteer computing, in order to form a Cloud system with the goal of providing services at little or no cost to end users. Section 2.3 introduced Desktop Clouds as a new type of Cloud with several challenges to be tackled before it represents a practical proposition.

High node volatility was identified as a research challenge affecting the outcomes of Desktop Cloud systems. Section 2.3.4.4 outlined that such issues can be resolved by developing a proper VM allocation mechanism. Section 3.2 reviewed the state-of-the-art methods proposed in the literature. However, this demonstrated that little attention has been paid to the issue of node failure in VM allocation mechanisms and that the literature falls short of providing a plausible VM allocation mechanism for Desktop Cloud systems in the presence of node failure. Therefore, in section 4.3 this research proposed an FT mechanism as a novel VM allocation mechanism able to cope with the issue of node failure. The FT mechanism was improved further by developing, in section 4.4, a novel migration policy to improve utilisation of PMs during runtime.

The aim of this research is to improve the outcomes of Desktop Cloud systems, mainly in terms of throughput. In order to achieve this goal, several research hypotheses were tested:

- Hypothesis H1: *The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds*
- Hypothesis H2: *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud system*
- Hypothesis H3: *Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with a decrease of 2% in the throughput outcome*

Three experiments were conducted to test these hypotheses:

- *Experiment I. The Impact of Node Failure*: to demonstrate that throughput, power consumption and availability metrics can evaluate the impact of node failures on the outcomes of Desktop Clouds.
- *Experiment II. Evaluation of the FT Mechanism*: to evaluate the FT mechanism in improving the throughput of a Desktop Cloud system compared to other VM mechanisms tested in the first experiment.
- *Experiment III. Utilisation-Based Migration Policy*: to find the utilisation threshold that achieves a trade-off between increasing resource utilisation and decreasing throughput.

The main contributions of this research are in developing a novel VM allocation mechanism that is tolerant of faults occurring in PMs in Desktop Cloud systems and enhancing it with a novel migration policy to improve resource utilisation. In addition, this study has provided an analysis of the number of node failures at any given time and proposed evaluation metrics for the outcomes of VM allocation mechanisms in the presence of node failure, developing a tool to simulate Desktop Cloud systems now available for use by other researchers. The following sections provide a summary of the study's findings and other contributions.

8.1.1 Node Failure

It was outlined in this research that PMs in Desktop Cloud systems are, for several reasons, quite volatile. PMs can leave the system without prior notification if they become busy with local tasks, moreover they are connected to the system via unreliable connections, increasing the risk of failure. This research analysed the number of failures in real-world failure traces in two Desktop Grid systems: NotreDame and SETI@home systems. Studying such systems gives an idea of the extent of node failure in Desktop Cloud systems, because both Desktop Grid and Desktop Clouds may employ the same infrastructure.

The challenge of node failure was investigated further by simulating Desktop Cloud systems and evaluating the impact of node failure. DesktopCloudSim is a simulation extension tool developed by this research to conduct experiments in a simulated Desktop Cloud system. The impact of node failure was assessed by answering the following research question:

What is the difference between Desktop Clouds and Traditional Clouds in terms of throughput, power consumption and availability?

This was answered by stating that node failure can affect Desktop Cloud systems by decreasing throughput, thus the experiment demonstrated the need to develop a new VM allocation mechanism able to improve throughput in the presence of node failure. It also proved that DesktopCloudSim, the extension tool developed by this research to simulate node failure, is able to simulate Desktop Cloud systems.

8.1.2 Evaluation Metrics

Throughput, power consumption and availability metrics were proposed to evaluate VM allocation mechanisms in Desktop Cloud systems. Experiment I was conducted to investigate this claim by evaluating four mechanisms: FCFS; Greedy; RoundRobin and Random. The experiment tested the following hypothesis:

The proposed metrics can be employed to evaluate the impact of node failure on Desktop Clouds.

In order to test it, the experiment addressed the following research questions:

Q1.1 What is the impact, if any, of node failure on throughput?

Q1.2 Which VM allocation mechanism yields the highest throughput of the tested mechanisms in Desktop Clouds?

Q1.3 Which VM allocation mechanism consumes the least power of the tested mechanisms in Desktop Clouds?

Q1.4 Which VM allocation mechanism yields the best availability of the tested mechanisms in Desktop Clouds?

Question *Q1.1* was answered by stating that node failure reduced throughput in both private and public Desktop Cloud systems, no matter which mechanism was employed. In answer to Question *Q1.2,* it was found that Greedy VM allocation is the mechanism yielding the highest throughput when employed in a private Desktop Cloud system. This mechanism was also the answer to Question *Q1.3,* being the one that consumed the least power, and indeed to *Q1.4,* as it yielded the greatest availability of PMs, when employed in a public Desktop Cloud system. By answering the above research questions the experiment demonstrated that metrics are able to evaluate the impact of node failure on a Desktop Cloud system and that there is a need to develop a novel VM mechanism that can cope with its issue of node failure.

8.1.3 Throughput Improvement

This research contributes to the field of Desktop Cloud computing by developing a novel VM allocation mechanism, termed the FT mechanism, able to improve throughput in the presence of node failure. The experiment tested the following, Hypothesis H2:

> *Employing a replication technique within the FT mechanism will improve the throughput of a Desktop Cloud System.*

In order to test this hypothesis, the research answered the following research questions:

> *Q2.1 What is the impact of employing the FT mechanism on the power consumed by nodes with a Desktop Cloud system?*
>
> *Q2.2 What is the impact of employing the FT mechanism on the availability of nodes in Desktop Clouds?*

The experiment demonstrated that the FT mechanism improved the throughput of both private and public Desktop Cloud systems statistically significantly more than traditional VM allocation mechanisms. However, the replication technique employed meant the improvement comes at a price, the consumption of considerable power. Question *Q2.1* was answered by saying that the FT mechanism led to greater energy consumption by PMs in Desktop Cloud systems than other mechanisms, therefore it is recommended that a technique is adopted to minimise this effect. In addition, the experiment revealed that availability was affected by using the replication technique, answering Question *Q2.2.*

8.1.4 Power Saving

Experiment II showed that the FT mechanism consumes a considerable amount of power, a result of little attention being directed at resource utilisation during runtime. This was improved by implementing a migration policy to improve resource utilisation with an acceptable degree of loss of throughput. Experiment III was conducted to assess the UBMP mechanism developed by this study to achieve the goal of the optimal trade-off between resource utilisation and decreased throughput. Hypothesis H3 was investigated:

Setting a utilisation threshold for online VM migration will reduce power consumption in the FT mechanism with an accepted decrease in throughput.

There are several related research questions that this experiment answered:

Q3.1 What is the most suitable node utilisation threshold to reduce consumption of power with minimal impact on throughput?
Q3.2 What is the impact of online VM migration policy on the throughput of a Cloud system?

Utilisation is used in this migration technique. If a PM has utilisation above zero but below a certain level, all VMs allocated to this PM will be migrated to another in order to improve utilisation. The experiment answered both questions by identifying a critical utilisation level of 40%. At this level there is optimal trade-off between throughput and power consumption. The experiment found that setting this threshold in UBMP statistically significantly reduced power consumption by about 20%, with the penalty of a maximum drop in throughput of 2%.

8.2 Future Work

The work carried out by this study can be extended into several promising areas in Desktop Clouds. First, several enhancements can be applied to the FT mechanism. Second, it can be investigated how to stimulate people to take part in Desktop Cloud systems. The last way is to look into the prospect of running applications on Desktop Cloud systems.

8.2.1 Mechanism Enhancement

One of the features of Desktop Cloud systems is that the PMs are widely distributed around the globe. This can be beneficial to the FT mechanism in reducing the impact on networks and improving response times. This direction of research can be extended further by developing a location-aware policy for the FT mechanism to select a PM to host a VM for a user in such a way that it is close as possible to the user. The same principle can be used in the UBMP policy to choose PMs for migration that are close to those from which VMs are migrating. Such a policy for the node selection and migration phases can help to reduce the overheads on the network.

Studies such as [172] propose approaches to predict node failure in Desktop Grids. This idea can be extended by the UBMP mechanism to trig the migration of VM instances from PMs just before they fail as it is predicated. This can be useful in two ways. The first advantage is that it will improve throughput and the second that it reduces the impact on the network, because there will be fewer new VM instances to create either primaries or replicas for failed VMs.

To improve prediction by node failure approaches, a technique based on experience may be employed by the FT mechanism. This can learn from experience in the short-term and long-term phases. For the former, it can observe the number of failures either to increase the number of required replicas, if the level of failures is quite high, or to destroy them if they are too few to reduce power. In the longer term, the technique can create a table to classify nodes according to their history of reliability. For example, a possible way is to assign nodes with a history of high reliability as primary nodes and those with low reliability as hosts to replicate VM instances.

Another step is to integrate the proposed mechanism into open-source Cloud middleware software, such as Eucalyptus. This can open up further avenues on which to undertake research into developing Cloud middleware suitable for Desktop Clouds and thus to conduct further experiments with real-world Desktop Cloud systems.

Another dimension to consider for future research is the number of running VM instances. The question of whether the number of running VM instances affects the outcomes of a Desktop Cloud system in the presence of node failure seems a feasible research question. Similarly, the issue about the relationship between the number of running VMs and that of the PMs available in a Desktop Cloud system might be raised, to improve the FT mechanism by employing a load-aware enhancement.

The UBMP policy showed that it can reduce power consumption with an acceptable level of throughput downgrade. However, a future goal could be to improve it by finding the optimum level of trade-off between resource utilisation and decrease in throughput. In addition, the number of migrations needed to be executed during runtime can have a negative impact on a network. Therefore, the UBMP can be investigated further by adding the new goal of minimising the number of migrations of VM instances. In short, a feasible future improvement to the UBMP is to find the optimum balance of improved resource utilisation and migration, along with minimal decrease in throughput.

8.2.2 Stimulate Contributors

An issue for further attention is how to motivate people to join Desktop Cloud systems as contributors or customers. There are some obvious reasons why people contribute their computing resources to research projects such as SETI@home, for instance for the sake of improving knowledge. It may be said that convincing people to contribute to Desktop Grids is much easier than getting them to contribute to Desktop Clouds. A contributor needs only to install a piece of software, BOINC software for instance, on a SETI@home system to run small batches of jobs, while it is more complicated to be a part of a Desktop Cloud, where people need to install virtualisation software. Some would be reluctant to do so, especially those with no experience in computer science.

A study might be carried out to ascertain public opinion on what would make people join Desktop Cloud systems. This survey would help to develop approaches to stimulate joining them, for instance by introducing a credit scheme for contributors: the more a contributor offers their available resources in a Desktop Cloud system, the more credit they accrue to be used to secure benefits from the system. For example, credit might be spent by the contributor to allow them high priority access to services.

The authors in [188] propose having a market for people to sell their computing resources over the Internet as part of a Desktop Cloud system. This idea opens up another research window on the potential to achieve this ambition. For example, a decision policy might be used to help people decide whether to sell their resources at a given time if the offered price is right. The policy can calculate the running cost of a PM, mainly power consumption, in order to assess whether it is profitable to join the Desktop Cloud system.

8.2.3 Application Type

According to Marosi et al., it is important for Desktop Cloud systems to deal with various types of applications [11]. However, an open issue for researchers to investigate is the kind of applications can be run on Desktop Cloud systems. Applications that require a rapid response time are not appropriate for such systems. In addition, the question can be raised about the ability of a Desktop Cloud to accommodate long-term issues such as web services in the presence of node failure.

The type of applications running on VM instances can monitored to assess the quality of service provided and to check that they are not undertaking malicious behaviour, which can provide an extra level of security. Researchers have promising opportunities to develop a means of monitoring Desktop Cloud systems, and to apply and test existing mechanisms developed for Traditional Clouds.

8.3 Final Remarks

Desktop Clouds are derived from the paradigm of Cloud computing, employing computing resources when they become idle in order to provide Cloud services at less expense. To work towards Desktop Clouds becoming a viable Cloud model, this research has focused on tackling the challenge of node failure. A novel failure-tolerant VM allocation mechanism was developed throughout that evaluated by throughput, power consumption and availability metrics through DesktopCloudSim, the simulation tool. The outcome is a VM allocation mechanism that can be employed in Desktop Cloud systems to reduce the impact of node failure. The mechanism was improved further by developing the UBMP mechanism, which improves resource utilisation in order to save energy. The mechanism can stimulate further innovation in Desktop Cloud computing so it represents a real alternative to Traditional Clouds.

List of References

[1] "MIT Centiminal." 1961.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in *Grid Computing Environments Workshop, GCE 2008*, 2008, pp. 1–10.

[3] S. S. L. Garfinkel, "An evaluation of amazon's grid computing services: EC2, S3, and SQS," *Cent.*, 2007.

[4] R. Buyya, R. Buyya, C. S. Yeo, C. S. Yeo, S. Venugopal, S. Venugopal, J. Broberg, J. Broberg, I. Brandic, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, p. 17, Jun. 2009.

[5] D. P. Anderson and G. Fedak, "The computational and storage potential of volunteer computing," in *Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006. CCGRID 06*, 2006, pp. 73–80.

[6] D. Kondo, G. Fedak, F. Cappello, A. a. Chien, and H. Casanova, "Characterizing resource availability in enterprise desktop grids," *Futur. Gener. Comput. Syst.*, vol. 23, no. 7, pp. 888–903, Aug. 2007.

[7] A. Gupta and L. K. L. Awasthi, "Peer enterprises: A viable alternative to Cloud computing?," in *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, 2009, vol. 2, pp. 1–6.

[8] R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson, *The Interaction of Parallel and Sequential Workloads on a Network of Workstations*, vol. 23, no. 1. ACM, 1995.

[9] J. B. Weissman, P. Sundarrajan, A. Gupta, M. Ryden, R. Nair, and A. Chandra, "Early experience with the distributed nebula cloud," in *Proceedings of the fourth international workshop on Data-intensive distributed computing*, 2011, pp. 17–26.

[10] H. N. Van, F. D. Tran, and J. M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," in *Proceedings - IEEE 9th International Conference on Computer and Information Technology, CIT 2009*, 2009, vol. 1, pp. 357–362.

[11] A. Marosi, J. Kovács, and P. Kacsuk, "Towards a volunteer cloud system," *Futur. Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1442–1451, Mar. 2013.

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*, 2009, pp. 124–131.

[13] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Comput.*, vol. 12, no. 1, pp. 1–15, Oct. 2008.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. - Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[15] G. Sakellari and G. Loukas, "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing," *Simul. Model. Pract. Theory*, vol. 39, pp. 92–103, Dec. 2013.

[16] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[17] D. De Roure, M. M. A. Baker, N. R. Jennings, N. R. Shadbolt, and D. De Roure, "The evolution of the Grid," in *Grid Computing*, 2003, pp. 65–100.

[18] I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," in *Second International Workshop, IPTPS 2003*, 2003, pp. 118–128.

[19] J. Yu and R. Buyya, "A taxonomy of workflow management systems for Grid computing," *J. Grid Comput.*, vol. 3, no. 3–4, pp. 171–200, Jan. 2005.

[20] I. Foster, "What is the grid? a three point checklist," *GRID today*, 2002.

[21] I. Foster, C. Kesselman, and S. Tuecke, "The open grid services architecture," in *The grid: blueprint for a new computing infrastucutre*, 2004, pp. 215 – 258.

[22] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. A. Chien, "Characterizing and evaluating desktop grids: an empirical study," *18th Int. Parallel Distrib. Process. Symp. 2004. Proceedings.*, no. C, 2004.

[23] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, "Characterizing and classifying desktop grid," in *Proceedings - Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2007*, 2007, pp. 743–748.

[24] D. Anderson, "BOINC: A System for Public Resource Computing and Storage," *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing*, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1382809. [Accessed: 27-Feb-2012].

[25] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11. pp. 56–61, 2002.

[26] B. Rood and M. J. Lewis, "Multi-state grid resource availability characterization," *2007 8th IEEE/ACM Int. Conf. Grid Comput.*, pp. 42–49, Sep. 2007.

[27] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a hunter of idle workstations," *[1988] Proceedings. 8th Int. Conf. Distrib.*, pp. 104–111, 1988.

[28] R. Alsoghayer and K. Djemame, “Resource failures risk assessment modelling in distributed environments,” *J. Syst. Softw.*, vol. 88, no. 1, pp. 42–53, 2014.

[29] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, “A View of Cloud Computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[30] J. Murchinson, “Google and IBM Announced University Initiative to Address Internet-Scale Computing Challenges,” 2007. [Online]. Available: http://www-03.ibm.com/press/en/pressrelease/22414.wss. [Accessed: 12-Jun-2015].

[31] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, “The characteristics of cloud computing,” in *Proceedings of the International Conference on Parallel Processing Workshops*, 2010, pp. 275–279.

[32] K. Jeffery and B. Neidecker-Lutz, “The Future of Cloud Computing Opportunities for European Cloud Computing Beyond 2010,” *Expert Gr. report, public version*, 2010.

[33] B. Rochwerger, D. Breitgand, E. Levy, a. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, “The Reservoir model and architecture for open federated cloud computing,” *IBM J. Res. Dev.*, vol. 53, no. 4, pp. 1–11, Jul. 2009.

[34] A. Chandra and J. Weissman, “Nebulas: Using distributed voluntary resources to build clouds,” in *Proceedings of the 2009 conference on Hot topics in cloud computing*, 2009, pp. 2–2.

[35] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, “Cloud Computing: a Perspective Study,” *New Gener. Comput.*, vol. 28, no. 2, pp. 137–146, Jun. 2010.

[36] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1. ACM, p. 50, 2008.

[37] M. E. Bégin, B. Jones, J. Casey, E. Laure, F. Grey, C. Loomis, and R. Kubli, “An EGEE Comparative Study Grids and Clouds - Evolution or Revolution?,” *EGEE III Proj. Rep.*, vol. 30, pp. 1–33, 2008.

[38] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, “Scientific cloud computing: Early definition and experience,” in *Proceedings - 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*, 2008, pp. 825–830.

[39] J. Geelan, “Twenty-one experts define cloud computing,” *Cloud Comput. J.*, 2009.

[40] P. Mell and T. Grance, “The NIST definition of cloud computing,” *Natl. Inst. Stand. Technol.*, vol. 53, no. 6, 2009.

[41] K. L. Kroeker, “The evolution of virtualization,” *Commun. ACM*, vol. 52, no. 3, pp. 18–20, 2009.

[42] T. Dillon, C. Wu, and E. Chang, “Cloud computing: Issues and challenges,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 27–33.

[43] B. P. Rimal, E. Choi, and I. Lumb, “A Taxonomy and Survey of Cloud Computing Systems,” *2009 Fifth Int. Jt. Conf. INC, IMS IDC*, pp. 44–51, Aug. 2009.

[44] M. Hammond, R. Hawtin, L. Gillam, and C. Oppenheim, “Cloud computing for research,” *Final Report. Curtis+ Cart. Consult. Ltd*, vol. 7, 2010.

[45] X. Chen, G. Wills, L. Gilbert, and D. Bacigalupo, “Using Cloud for Research: A Technical Review,” *JISC Final Rep.*, 2010.

[46] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, Apr. 2010.

[47] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, “Analysis of Virtualization Technologies for High Performance Computing Environments,” *2011 IEEE 4th Int. Conf. Cloud Comput.*, pp. 9–16, Jul. 2011.

[48] A. Iosup, R. Prodan, and D. Epema, “IaaS Cloud Benchmarking: Approaches, Challenges, and Experience,” *Proc. ACM/IEEE Conf. …*.

[49] T. Ristenpart, E. Tromer, S. Savage, and H. Shacham, “Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds,” in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 199–212.

[50] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Prentice Hall, 2005.

[51] C. Schroth and T. Janner, “Web 2.0 and soa: Converging concepts enabling the internet of services,” *IT Prof.*, vol. 9, no. 3, pp. 36–41, 2007.

[52] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, “Volunteer computing and desktop cloud: The cloud @ home paradigm,” in *Proceedings - 2009 8th IEEE International Symposium on Network Computing and Applications, NCA 2009*, 2009, pp. 134–139.

[53] F. Lui, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badget, and D. Leaf, “NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology,” *Nist Spec. Publ. 500-292*, 2011.

[54] G. Kirby, A. Dearle, A. Macdonald, and A. Fernandes, “An Approach to Ad hoc Cloud Computing,” *Arxiv Prepr. arXiv1002.4738*, 2010.

[55] A. Andrzejak, D. Kondo, and D. P. Anderson, “Exploiting non-dedicated resources for cloud computing,” in *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, 2010, pp. 341–348.

[56] K. Graffi, D. Stingl, C. Gross, H. Nguyen, A. Kovacevic, and R. Steinmetz, “Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems,” *cs.uni-paderborn.de*, pp. 27–34, 2010.

[57] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, “Cloud@Home: Bridging the gap between volunteer and cloud computing,” in *Emerging Intelligent Computing Technology and Applications*, 2009, vol. 5754 LNCS, pp. 423–432.

[58] A. Harutyunyan, J. Blomer, P. Buncic, I. Charalampidis, F. Grey, A. Karneyeu, D. Larsen, D. Lombraña González, J. Lisec, B. Segal, and P. Skands, “CernVM Co-Pilot: an Extensible Framework for Building Scalable Computing Infrastructures on the Cloud,” *J. Phys. Conf. Ser.*, vol. 396, no. 3, p. 032054, Dec. 2012.

[59] M. Christodorescu and R. Sailer, “Cloud security is not (just) virtualization security: a short paper,” *... Comput. Secur.*, pp. 97–102, 2009.

[60] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, “Cloud@Home: Bridging the gap between volunteer and cloud computing,” in *Emerging Intelligent Computing Technology and Applications*, 2009, vol. 5754 LNCS, pp. 423–432.

[61] S. Zhang, X. Chen, and X. Huo, “The comparison between cloud computing and grid computing,” in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, 2010, vol. 11, no. 46, pp. V11–72.

[62] A. Redolfi, R. McClatchey, A. Anjum, A. Zijdenbos, D. Manset, F. Barkhof, C. Spenger, Y. Legré, L.-O. Wahlund, C. B. di San Pietro, and G. B. Frisoni, “Grid infrastructures for computational neuroscience: the neuGRID example,” *Future Neurol.*, vol. 4, no. 6, pp. 703–722, Nov. 2009.

[63] D. Talia, “The open grid services architecture: where the grid meets the web,” *Internet Comput. IEEE*, vol. 6, no. 6, pp. 67–71, 2002.

[64] D. A. Menasce and E. Casalicchio, “QoS in grid computing,” *Internet Comput. IEEE*, vol. 8, no. 4, pp. 85–87, 2004.

[65] A. A. R. Butt, S. Adabala, N. N. H. Kapadia, Ã. S. Adabala, and R. J. Figueiredo, “Grid-computing portals and security issues,” *J. Parallel*, vol. 63, no. 10, pp. 1006–1014, Oct. 2003.

[66] D. Kondo, A. Chien, and H. Casanova, “Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids,” *Proc. ACM/IEEE SC2004 Conf.*, 2004.

[67] O. Curran, P. Downes, J. Cunniffe, A. Shearer, and J. P. Morrison, “Resource Aggregation and Workflow with Webcom,” *High Perform. Comput. Commun.*, vol. 4782, pp. 108–119, 2007.

[68] S. Min and J. Holliday, “Super-Peer Availability Prediction Strategy in Unstructured P2P Network,” in *Network and Parallel Computing, 2009. NPC’09. Sixth IFIP International Conference on*, 2009, vol. 0, pp. 23–29.

[69] S. C. S. Choi, M. B. M. Baik, C. H. C. Hwang, J. G. J. Gil, and H. Y. H. Yu, “Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment,” *Third IEEE Int. Symp. Netw. Comput. Appl. 2004. (NCA 2004). Proceedings.*, pp. 0–5, 2004.

[70] K. Popovic and Z. Hocenski, “Cloud computing security issues and challenges,” in *MIPRO, 2010 Proceedings of the 33rd International Convention*, 2010, pp. 344–349.

[71] X. Jing and Z. Jian-jun, “A Brief Survey on the Security Model of Cloud Computing,” *2010 Ninth Int. Symp. Distrib. Comput. Appl. to Business, Eng. Sci.*, pp. 475–478, Aug. 2010.

[72] P. Goyal, “Enterprise Usability of Cloud Computing Environments: Issues and Challenges,” *2010 19th IEEE Int. Work. Enabling Technol. Infrastructures Collab. Enterp.*, pp. 54–59, 2010.

[73] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, “Cloud Computing – A Classification, Business Models, and Research Directions,” *Bus. Inf. Syst. Eng.*, vol. 1, no. 5, pp. 391–399, Sep. 2009.

[74] A. Khajeh-hosseini and I. Sommerville, “Research challenges for enterprise cloud computing,” *Arxiv Prepr. arXiv*, 2010.

[75] D. Durkee, “Why cloud computing will never be free,” *Commun. ACM*, vol. 53, no. 5, p. 62, May 2010.

[76] M. Hauck, M. Huber, M. Klems, S. Kounev, J. M\"uller-Quade, A. Pretschner, R. Reussner, and S. Tai, “Challenges and Opportunities of Cloud Computing,” *pp.info.uni-karlsruhe.de*, 2010.

[77] B. Addis, D. Ardagna, B. Panicucci, and L. Zhang, “Autonomic Management of Cloud Service Centers with Availability Guarantees,” *2010 IEEE 3rd Int. Conf. Cloud Comput.*, pp. 220–227, Jul. 2010.

[78] S. Distefano, A. Puliafito, M. Rak, S. Venticinque, U. Villano, A. Cuomo, G. Di Modica, and O. Tomarchio, “QoS management in Cloud@Home infrastructures,” in *Proceedings - 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2011*, 2011, pp. 190–197.

[79] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, “FTCloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications,” *2010 IEEE 21st Int. Symp. Softw. Reliab. Eng.*, pp. 398–407, Nov. 2010.

[80] B. Q. Cao, B. Li, and Q. M. Xia, “A Service-Oriented Qos-Assured and Multi-Agent Cloud Computing Architecture,” *Cloud Comput.*, pp. 644–649, 2009.

[81] A. Andrzejak, D. Kondo, and D. P. Anderson, “Ensuring collective availability in volatile resource pools via forecasting,” in *19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2008, Samos Island, Greece, September 22-26, 2008. Proceedings*, 2008, no. contract 34084, pp. 149–161.

[82] P. Endo, A. de A. Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs, “Resource allocation for distributed cloud: concepts and research challenges,” *Network, IEEE*, vol. 25, no. August, pp. 42–46, 2011.

[83] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, “Cloud monitoring: A survey,” *Computer Networks*, vol. 57, no. 9. Elsevier B.V., pp. 2093–2115, 2013.

[84] A. Marinos and G. Briscoe, “Community cloud computing,” in *Cloud Computing (First International Conference, CloudCom 2009)*, 2009, vol. 5931 LNCS, pp. 472–484.

[85] D. Nurmi, R. Wolski, and C. Grzegorczyk, “Eucalyptus : A Technical Report on an Elastic Utility Computing Archietcture Linking Your Programs to Useful Systems,” *... Tech. Rep.*, 2008.

[86] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[87] K. Scarfone, *Guide to Security for Full Virtualization Technologies*, vol. 125. DIANE Publishing, 2001.

[88] W. Shi and B. Hong, “Towards Profitable Virtual Machine Placement in the Data Center,” *2011 Fourth IEEE Int. Conf. Util. Cloud Comput.*, pp. 138–145, Dec. 2011.

[89] A. Beloglazov and R. Buyya, “Energy efficient resource management in virtualized cloud data centers,” in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826–831.

[90] M. Mishra and A. Das, “Dynamic resource management using virtual machine migrations,” *IEEE Commun. Mag.*, no. September, pp. 34–40, 2012.

[91] X. Meng, C. Isci, J. O. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, “Efficient resource provisioning in compute clouds via VM multiplexing,” *Proceeding 7th Int. Conf. Auton. Comput. - ICAC’10*, p. 11, 2010.

[92] A. Corradi, M. Fanelli, and L. Foschini, “VM consolidation: A real case based on OpenStack Cloud,” *Futur. Gener. Comput. Syst.*, vol. 32, no. 1, pp. 118–127, Jun. 2014.

[93] R. Wasim Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, “A survey on virtual machine migration and server consolidation techniques for cloud data centers,” *J. Netw. Comput. Appl.*, vol. 52, pp. 11–25, 2015.

[94] D. Kapil, E. S. Pilli, and R. C. Joshi, “Live virtual machine migration techniques: Survey and research challenges,” *Proc. 2013 3rd IEEE Int. Adv. Comput. Conf. IACC 2013*, pp. 963–969, 2013.

[95] A. Kivity, U. Lublin, A. Liguori, Y. Kamay, and D. Laor, “kvm: the Linux virtual machine monitor,” *Proc. Linux Symp.*, vol. 1, pp. 225–230, 2007.

[96] A. Verma, P. Ahuja, and A. Neogi, “pMapper: power and migration cost aware application placement in virtualized systems,” in *Middleware ’08 Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, 2008, pp. 243–264.

[97] R. N. Calheiros, R. Buyya, and C. a F. De Rose, “A heuristic for mapping virtual machines and links in emulation testbeds,” *Proc. Int. Conf. Parallel Process.*, pp. 518–525, 2009.

[98] K. Mills, J. Filliben, and C. Dabrowski, “Comparing VM-Placement Algorithms for On-Demand Clouds,” *2011 IEEE Third Int. Conf. Cloud Comput. Technol. Sci.*, pp. 91–98, Nov. 2011.

[99] J. T. Piao and J. Yan, “A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing,” *2010 Ninth Int. Conf. Grid Cloud Comput.*, pp. 87–92, Nov. 2010.

[100] F. Chang, J. Ren, and R. Viswanathan, “Optimal Resource Allocation in Clouds,” *2010 IEEE 3rd Int. Conf. Cloud Comput.*, pp. 418–425, Jul. 2010.

[101] N. Antonopoulos and L. Gillam, *Cloud Computing: Principles, Systems and Applications*. London: Springer, 2010.

[102] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, “SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter,” *J. Netw. Comput. Appl.*, vol. 45, pp. 108–120, 2014.

[103] W. Voorsluys, S. K. Garg, and R. Buyya, “Provisioning spot market cloud resources to create cost-effective virtual clusters,” in *11th International Conference, ICA3PP*, 2011, no. PART 1, pp. 395–408.

[104] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing Principles and Paradigms*. 2011.

[105] M. Garey and D. Johnson, “A 71/60 theorem for bin packing,” *J. Complex.*, vol. 106, pp. 65–106, 1985.

[106] S. Srikantaiah, A. Kansal, and F. Zhao, “Energy aware consolidation for cloud computing,” in *HotPower’08: Proceedings of the 2008 conference on Power aware computing and systems*, 2008.

[107] D. Jiang, P. Huang, P. Lin, and J. Jiang, “Energy efficient VM placement heuristic algorithms comparison for cloud with multidimensional resources,” in *Third International Conference, ICICA 2012 Proceedings*, 2012, vol. 7473 LNCS, pp. 413–420.

[108] B. Sotomayor, R. R. S. Montero, I. M. Llorente, and I. Foster, “Virtual infrastructure management in private and hybrid clouds,” *IEEE Internet Comput.*, vol. 13, no. 5, pp. 14–22, Sep. 2009.

[109] A. Nathani, S. Chaudhary, and G. Somani, “Policy based resource allocation in IaaS cloud,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 1, pp. 94–103, Jan. 2012.

[110] U. Schwiegelshohn and R. Yahyapour, “Analysis of first-come-first-serve parallel job scheduling,” *Proc. ninth Annu. ACM …*, pp. 629–638, 1998.

[111] P. Sempolinski and D. Thain, “A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus,” *2010 IEEE Second Int. Conf. Cloud Comput. Technol. Sci.*, pp. 417–426, Nov. 2010.

[112] B. Korte and L. Lovász, “Mathematical structures underlying greedy algorithms BT - Fundamentals of Computation Theory,” *Fundam. Comput. Theory*, vol. 117, pp. 205–209, 1981.

[113] E. L. Hahne, “Round robin scheduling for fair flow control in data communication networks,” 1987.

[114] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, vol. 4, no. 1. 2009.

[115] P. Graubner, M. Schmidt, and B. Freisleben, “Energy-efficient management of virtual machines in Eucalyptus,” in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 243–250.

[116] R. Buyya, A. Beloglazov, and J. Abawajy, “Energy-Efficient Management of Data Center Resources for Cloud Computing : A Vision , Architectural Elements , and Open Challenges,” *arXiv Prepr. arXiv1006.0308*, no. Vm, pp. 1–12, 2010.

[117] V. Vazirani, *Approximation algorithms*, Second. New York, New York, USA: Springer, 2003.

[118] E. Feller, C. Rohr, D. Margery, and C. Morin, “Energy management in IaaS clouds: A holistic approach,” in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 204–212.

[119] X. Liao, H. Jin, and H. Liu, “Towards a green cluster through dynamic remapping of virtual machines,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 2, pp. 469–477, Feb. 2012.

[120] K. Tsakalozos, M. Roussopoulos, and A. Delis, “VM placement in non-homogeneous IaaS-clouds,” in *Service-Oriented Computing*, vol. 7084 LNCS, G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad, Eds. Springer Berlin Heidelberg, 2011, pp. 172–187.

[121] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, “VM placement strategies for cloud scenarios,” in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 852–859.

[122] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” in *Proceedings - IEEE INFOCOM*, 2010.

[123] B. Palanisamy, A. Singh, L. Liu, and B. Jain, “Purlieus: Locality-aware resource allocation for MapReduce in a cloud,” *2011 Int. Conf. High Perform. Comput. Networking, Storage Anal.*, pp. 1–11, 2011.

[124] J. Dean and S. Ghemawat, “MapReduce: Simplied Data Processing on Large Clusters,” in *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, 2004, vol. 103, no. 34, pp. 137–149.

[125] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, “A stable network-aware VM placement for cloud systems,” in *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 2012, pp. 498–506.

[126] F. P. Tso, G. Hamilton, K. Oikonomou, and D. P. Pezaros, “Implementing scalable, network-aware virtual machine migration for cloud data centers,” in *IEEE International Conference on Cloud Computing, CLOUD*, 2013, pp. 557–564.

[127] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou, “Efficient task replication and management for adaptive fault tolerance in Mobile Grid environments,” *Futur. Gener. Comput. Syst.*, vol. 23, no. 2, pp. 163–178, 2007.

[128] H. L. H. Lee, D. P. D. Park, M. H. M. Hong, S.-S. Y. S.-S. Yeo, S. K. S. Kim, and S. K. S. Kim, “A Resource Management System for Fault Tolerance in Grid Computing,” *2009 Int. Conf. Comput. Sci. Eng.*, vol. 2, pp. 609–614, 2009.

[129] F. Machida, M. Kawato, and Y. Maeno, “Redundant virtual machine placement for fault-tolerant consolidated server clusters,” in *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, 2010, pp. 32–39.

[130] Y. Zhang, Z. Zheng, and M. R. Lyu, “BFTCloud: A Byzantine Fault Tolerance Framework for Voluntary-Resource Cloud Computing,” *2011 IEEE 4th Int. Conf. Cloud Comput.*, pp. 444–451, Jul. 2011.

[131] B. Javadi, D. Kondo, J. Vincent, and D. P. Anderson, “Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1896–1903, 2011.

[132] B. Javadi, J. Abawajy, and R. Buyya, “Failure-aware resource provisioning for hybrid Cloud infrastructure,” *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1318–1331, Oct. 2012.

[133] “OpenStack.” [Online]. Available: http://www.openstack.org/.

[134] A. Oliner, L. Rudolph, and R. Sahoo, “Cooperative checkpointing theory,” in *20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, 2006, vol. 2006, pp. 14–23.

[135] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, “Remus: High availability via asynchronous virtual machine replication,” *NSDI’08 Proc. 5th USENIX Symp. Networked Syst. Des. Implement.*, vol. vi, pp. 161–174, 2008.

[136] R. W. Ahmad, A. Gani, S. H. Ab. Hamid, M. Shiraz, F. Xia, and S. a. Madani, “Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues,” *J. Supercomput.*, 2015.

[137] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, “Cost of virtual machine live migration in clouds: A performance evaluation,” *Proc. 1st Int. Conf. Cloud Comput. CloudCom 2009*, pp. 254–265, 2009.

[138] P. Barham, B. Dragovic, K. Fraser, and S. Hand, “Xen and the art of virtualization,” *ACM SIGOPS*, 2003.

[139] Z. Li, L. O’Brien, H. Zhang, and R. Cai, “On a Catalogue of Metrics for Evaluating Commercial Cloud Services,” *... Int. Conf.*, pp. 164–173, 2012.

[140] Í. Goiri, F. Julià, J. O. Fitó, M. Macías, and J. Guitart, “Supporting CPU-based guarantees in cloud SLAs via resource-level QoS metrics,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 8, pp. 1295–1302, Oct. 2012.

[141] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, “What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-Service Offerings,” *2011 IEEE 4th Int. Conf. Cloud Comput.*, pp. 484–491, Jul. 2011.

[142] V. Stantchev, “Performance Evaluation of Cloud Computing Offerings,” *2009 Third Int. Conf. Adv. Eng. Comput. Appl. Sci.*, pp. 187–192, Oct. 2009.

[143] D. Villegas, A. Antoniou, S. M. Sadjadi, and A. Iosup, “An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds,” *2012 12th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. (ccgrid 2012)*, vol. 2, no. Section III, pp. 612–619, May 2012.

[144] H. N. V. H. N. Van, F. D. Tran, and J.-M. Menaud, “Performance and Power Management for Cloud Infrastructures,” *Cloud Comput. (CLOUD), 2010 IEEE 3rd Int. Conf.*, pp. 329–336, Jul. 2010.

[145] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing Principles and Paradigms*. John Wiley & Sons, 2010.

[146] S. K. Garg, S. Versteeg, and R. Buyya, “A framework for ranking of cloud computing services,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023, Jun. 2013.

[147] C. Bash, T. Cader, Y. Chen, D. Gmach, R. Kaufman, D. Milojicic, A. Shah, and P. Sharma, “Cloud Sustainability Dashboard, Dynamically Assessing Sustainability of Data Centers and Clouds,” in *Proceedings of the fifth open cirrus summit*, 2011.

[148] K. D. Lange, “Identifying shades of green: The SPECpower benchmarks,” *Computer (Long. Beach. Calif).*, vol. 42, no. 3, pp. 95–97, 2009.

[149] Y. C. Lee and A. Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” *J. Supercomput.*, vol. 60, no. 2, pp. 268–280, Mar. 2010.

[150] J. Sokolowski and C. Banks, *Principles of Modeling and Simulation: A Multidisciplinary Approach*. Hoboken: John Wiley & Sons, 2009.

[151] S. Robinson, *Simulation: The Practice of Model Development and Use*. John Wiley & Sons, 2004.

[152] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities,” *High Perform. Comput. Simulation, 2009. HPCS’09*, pp. 1–11, Jun. 2009.

[153] J. Banks, J. Carson, B. L. Nelson, and D. Nicol, *Discrete-Event System Simulation*. London: Pearson Prentice Hall, 2005.

[154] R. Buyya and M. Murshed, “Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing,” *Concurr. Comput. Pract. …*, vol. 14, no. 13–15, pp. 1175–1220, Nov. 2003.

[155] F. Howell and R. McNab, “SimJava: A discrete event simulation library for java,” *Simul. Ser.*, 1998.

[156] H. Casanova, “Simgrid: a toolkit for the simulation of application scheduling,” *Proc. First IEEE/ACM Int. Symp. Clust. Comput. Grid*, pp. 430–437, 2001.

[157] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, “GroudSim: an event-based simulation framework for computational grids and clouds,” *Euro-Par 2010 Parallel …*, no. 261585, pp. 305–313, 2011.

[158] S. H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, “MDCSim: A multi-tier data center simulation platform,” in *Cluster Computing and Workshops, 2009. CLUSTER ’09. IEEE International Conference on*, 2009.

[159] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, “GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers,” in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010, vol. 62, no. 3, pp. 1–5.

[160] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, “ICanCloud: A Flexible and Scalable Cloud Infrastructure Simulator,” *J. Grid Comput.*, vol. 10, no. 1, pp. 185–209, Apr. 2012.

[161] K. and others McCanne, S. and Floyd, S. and Fall, K. and Varadhan, “Network simulator ns-2.” 1997.

[162] A. Núñez, J. Fernández, J. D. Garcia, L. Prada, and J. Carretero, “SIMCAN : A SIMulator Framework for Computer Architectures and Storage Networks,” in *Simutools ’08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008.

[163] M. Bux and U. Leser, “DynamicCloudSim: simulating heterogeneity in computational clouds,” *SWEET ’13 Proc. 2nd ACM SIGMOD Work. Scalable Work. Exec. Engines Technol.*, 2013.

[164] W. Chen and M. Rey, “WorkflowSim : A Toolkit for Simulating Scientific Workflows in Distributed Environments,” 2012.

[165] S. K. Garg and R. Buyya, “NetworkCloudSim: Modelling parallel applications in cloud simulations,” in *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing, UCC 2011*, 2011, no. Vm, pp. 105–113.

[166] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, “FederatedCloudSim: a SLA-aware federated cloud simulation framework,” in *Proceedings of the 2nd International Workshop on CrossCloud Systems - CCB ’14*, 2014, pp. 1–5.

[167] R. Buyya, R. Ranjan, and R. N. Calheiros, “InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services,” in *Algorithms and architectures for parallel processing*, 2010, vol. 6081 LNCS, no. PART 1, pp. 13–31.

[168] Í. Goiri, J. Guitart, and J. Torres, “Characterizing cloud federation for enhancing providers’ profit,” in *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, 2010, pp. 123–130.

[169] A. Field, *Discovering statistics using SPSS*, Third. SAGE Publications Ltd, 2009.

[170] B. Javadi, D. Kondo, A. Iosup, and D. Epema, “The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems,” *J. Parallel Distrib. Comput.*, vol. 73, no. 8, pp. 1208–1223, Aug. 2013.

[171] C. Reiss, J. Wilkes, and J. Hellerstein, “Google cluster-usage traces: format+ schema,” 2011.

[172] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, “Mining for statistical models of availability in large-scale distributed systems: An empirical study of SETI@home,” *2009 IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, pp. 1 – 10, 2009.

[173] F. Nadeem, R. Prodan, and T. Fahringer, “Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-purpose Grid,” *2008 Eighth IEEE Int. Symp. Clust. Comput. Grid*, pp. 348–357, May 2008.

[174] A. Iosup, M. Jan, O. Sonmez, and D. H. J. Epema, “On the dynamic resource availability in grids,” *2007 8th IEEE/ACM Int. Conf. Grid Comput.*, pp. 26–33, 2007.

[175] B. Schroeder and G. a Gibson, “A Large-Scale Study of Failures in High-Performance Computing Systems,” *IEEE Trans. Dependable Secur. Comput.*, vol. 7, no. 4, pp. 337–350, Oct. 2010.

[176] K. Park and V. S. Pai, “CoMon: a mostly-scalable monitoring system for PlanetLab,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006.

[177] DMTF, “Open Virtualization Format Specification,” 2010.

[178] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, “A proportional share resource allocation algorithm for real-time, time-shared systems,” *17th IEEE Real-Time Syst. Symp.*, pp. 288–299, 1996.

[179] K. V. Vishwanath and N. Nagappan, “Characterizing Cloud Computing Hardware Reliability,” in *Proceedings of the 1st ACM symposium on Cloud computing - SoCC ’10*, 2010, p. 193.

[180] B. M. Segal, P. Buncic, D. G. Quintas, D. L. Gonzalez, A. Harutyunyan, J. Rantala, and D. Weir, “Building a volunteer cloud,” *Memorias la ULA*, 2009.

[181] R. K. Sahoo, M. S. Squillante, and A. Sivasubramaniam, “Failure data analysis of a large-scale heterogeneous server environment,” in *International Conference on Dependable Systems and Networks, 2004*, 2004, pp. 772–781.

[182] O. Beaumont, L. Eyraud-Dubois, C. Thraves Caro, and H. Rejeb, “Heterogeneous resource allocation under degree constraints,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 926–937, 2013.

[183] B. Jennings and R. Stadler, “Resource Management in Clouds: Survey and Research Challenges,” *Journal of Network and Systems Management*, pp. 1–53, 2014.

[184] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, “Applying software engineering principles for designing Cloud@Home,” in *CCGrid 2010 - 10th*

IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing, 2010, pp. 618–624.

[185] Y. Shi, X. Jiang, and K. Ye, “An Energy-Efficient Scheme for Cloud Resource Provisioning Based on CloudSim,” *2011 IEEE Int. Conf. Clust. Comput.*, pp. 595–599, Sep. 2011.

[186] D. P. Anderson, “Volunteer computing,” *Crossroads*, vol. 16, no. 3, pp. 7–10, Mar. 2010.

[187] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, “Cost-benefit analysis of Cloud Computing versus desktop grids,” *2009 IEEE Int. Symp. Parallel Distrib. Process.*, 2009.

[188] K. Chard, K. Bubendorfer, S. Caton, and O. F. Rana, “Social cloud computing: A vision for socially motivated resource sharing,” *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 551–563, 2012.