ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Analysis of Availability Using Virtual Machine Placement Algorithms for Cloud IaaS

By

Ammanuel Shawel

Advisor

Dr. Mesfin Kifle

Submitted in partial fulfillment of the requirements of the degree of Masters of Science

in Telecommunication Engineering

February 2020

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Analysis of Availability Using Virtual Machine Placement Algorithms for Cloud IaaS

By

Ammanuel Shawel

Approved by:

_____	_____	_____
Chairman, School Graduate Committee	Signature	Date
Dr. Mesfin Kifle	_____	_____
Advisor	Signature	Date
_____	_____	_____
Examiner	Signature	Date
_____	_____	_____
Examiner	Signature	Date

Declaration

I, the undersigned, declare that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other university, and all sources and materials used for the thesis have been acknowledged.

Ammanuel Shawel	_____	_____
Author	Signature	Date

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Mesfin Kifle	_____	_____
Advisor	Signature	Date

Abstract

Cloud Computing paradigm is most popular for its flexibility in resource provisioning by creating virtual machines of the requested specification on the underlying physical infrastructure. Infrastructure-as-a-Service, which requires a provider to allocate virtual machines (VMs), has significant impact on the availability of accommodated applications. Due to this, several algorithms have been proposed for the VM placement problem with little or no objective comparison hiding the fact which one works best or what factors influence the availability of the algorithms. In this thesis work, comparison of four algorithms using metrics of availability is presented. The findings showed that the impact of VM placement algorithms along with memory and CPU utilization play an important role in maintaining the availability of IaaS for cloud. As a result, algorithms which perform well on one metric perform poorly on the other metrics underlining the importance of objectively comparing the availability of competing algorithms and highlighting the importance of thorough empirical studies not only for power consumption and other QoS attributes but also for availability as one important aspect of cloud computing.

Keywords Cloud IaaS availability, virtual machine placement, bin packing, heuristics

Acknowledgment

First and foremost, I give thanks to God for the strength, wisdom and grace to do this thesis work. Without Him this would not have been possible. I am grateful to my company ethio telecom and School of Electrical and Computer Engineering of Addis Ababa University, for making it possible for me to study here. I would also like to thank my advisor Dr. Mesfin Kifle for providing guidance and suggestions in my research and writing of this thesis. Last but not least, I am very grateful to my spouse and my children for their unconditional love and support to the completion of my work.

Table of Contents

List of Figures

List of Tables

Acronyms

BFD	Best Fit Decreasing
CL	Confidence Level
CPU	Central Processing Unit
CV	Coefficient of Variation
DC	Data Center
FFD	First Fit Decreasing
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
K-W	Kruskal-Wallis
MIPS	Minutes of Instruction per Second
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
P-Value	Probability Value
PE	Processing Element
PaaS	Platform as a Service
PM	Physical Machine
RAM	Random Access Memory
SLA	Service Level Agreement
SLO	Service Level Objective
SaaS	Software as a Service
VM	Virtual Machine
VMM	Virtual Machine Monitor

Chapter 1 Introduction

1.1 Background

The purpose of this section is to provide background information about cloud computing and its major enabling technologies, availability within the context of cloud and virtual machine placement algorithms in relation to cloud virtualization.

1.1.1 Cloud computing

Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service level agreement. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction[1].

Depending on the relationship between the provider and the consumer, a cloud can be classified into four deployment models: public, private, community and hybrid cloud. Public cloud is owned and operated by independent vendors and accessible to the general public. Private cloud is an internal utilization of cloud technologies which is maintained in-house and solely accessible to internal users within an organization. Community cloud is shared by several organizations and supports a specific community that has shared concerns. It may be managed by the organizations or a third party and may exist on-premise or off-premise. Hybrid cloud is a combination of two or more types of clouds (private, community, or public).

Cloud computing has three main service models. These are Software as a Service(SaaS), Platform as a Service(PaaS) and Infrastructure as a Service(IaaS). SaaS consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities apart from limited user-specific application configuration settings. PaaS provides capability to a consumer to deploy in the cloud infrastructure applications, libraries, services, and tools supported and no control on the underlying cloud infrastructure

including network, servers, operating systems, or storage. IaaS provides capabilities to the consumer to manage processing, storage, networks, and computing resources where the consumer can deploy and run arbitrary software[2].

1.1.2 Cloud Availability

Based on 2017 Computer Reseller News (CRN) report of cloud service failure, cloud service providers such as IBM, GitLab, Facebook and Amazon caused production data loss and prevented customers from accessing critical data for long hours causing a credibility loss on availability [29]. Availability is a non-functional requirement defined as the percentage of time a system or service is accessible[3]. This percentage determines the acceptable total outage time for any given period. It is agreed that availability is among the main challenges of cloud computing as more critical services are shifting towards this paradigm as reported by Armbrust et al. [4]. Individual services and their interactions between multiple components and automated services may cause problems in the cloud by increasing outage time. Availability in most research papers is related to other characteristics of the cloud such as performance, scalability, elasticity and security without making a proper distinction and proposing a comprehensive solution that would guarantee it in the cloud.

Cloud providers consider availability within the context of a Service Level Agreement (SLA) while research literature definitions are tailored to a proposed solution. Information Technology Infrastructure Library (ITIL) on the other hand, defines availability as the ratio of elapsed time between service getting up and down to the total of elapsed time to repair a configuration item or IT service plus the elapsed time between service getting up and down.

1.1.3 Cloud Enabling Technologies

The two major cloud-enabling technologies are virtualization and load balancing. Virtualization plays a major role in cloud computing as it provides virtual storage and computing services to the cloud clients. It is a methodology of dividing the resources of a computer into multiple execution environments by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others[5]. Though the primary focus for virtualization continues to be on servers,

virtualizing storage and networks is also emerging as a general strategy. Load balancing helps in the fair allocation of computing resources to achieve a high level of user satisfaction and proper use of resources. It is performed at two levels in cloud computing. At the virtual machine level, mapping is made between applications and virtual machines while at the host level mapping is made between the virtual machine and host resources.

1.1.4 Virtual Machine Placement

Cloud computing applies virtualization in which hardware resources of one or more computer systems are divided into several execution environments called Virtual Machines (VMs). Each VM is isolated from other VMs and can act as a complete system to execute the user applications. Hosting a VM on a Physical Machine (PM) or server involves the provisioning of required resources such as CPU, memory, storage and bandwidth. Inside a server PM, the VMs are controlled by a layer of software called VM Monitor (VMM) or hypervisor that resides between the hardware platform and the VMs. Generally, VMM supports the creating, migrating and terminating of VM instances [6].

VM migration is a cloud computing feature aimed to respond to the dynamic requests of the VMs in order to guarantee the promised Service Level Agreement (SLA) to the cloud consumers [7]. The VM migration process consists of selecting overloaded or underloaded physical machine, selecting VMs for migration and performing VM placement by utilizing a placement algorithm. VM placement is a critical operation which is conducted to determine the most appropriate PM or server to host the VM. Selecting a suitable host is very important to improve power efficiency, resource utilization and QoS aspects such as availability and performance in supporting cloud computing environment. Heuristics such as First-Fit, First-Fit-Decreasing, Best-Fit, Best-Fit-Decreasing, Next-Fit, Round-Robin and Worst-Fit accomplish results useful for dynamic VM placement where the demand is highly variable and will always generate a good solution in a considerable amount of time with physical machines. Consequently, an ideal VM placement scheme reduces the need for future VM migrations and improve the resource utilization and availability of IaaS in the data centers.

Making a comparison with respect to availability between VM allocation algorithms has obstacles. Due to the necessity of real-world test environments, workload traces and insufficient methodology, unlike some of the more mature fields of computing, algorithms already proposed have no way to tell how they can be compared to each other in terms of availability quality.

1.2 Motivation

In Ethiopia, enterprises up till now are far from adopting cloud computing as an IT provisioning method due to lack of awareness of the advantages and concern with the challenges. But recent endeavors by the government have shaded some light on its commitment to utilize the benefits of cloud in which ethiotelecom is on the forefront. Hence, this research is motivated by the future propensity of cloud computing in the country as well as the importance of service availability in the IaaS layer that greatly influence the QoS in its adoption.

1.3 Statement of the Problem

The dynamic changes of the load of a VM necessitate optimal VMs to PMs allocation. Data Center (DC) operators regularly re-optimize the mapping of VMs to PMs and perform the necessary migrations to get to the newly determined placement ultimately satisfying SLOs. The algorithm used by an operator for re-optimizing the placement of VMs has a large impact on multiple vital metrics such as energy consumption in consolidating the VMs to as few PMs as possible, application performance in avoiding SLO violations and migration [8]. VM placement is a bin-packing problem that is strongly NP-hard making the existence of an efficient exact algorithm very unlikely [9]. But due to the need to solve the problem of VMs to PMs placement in a reasonable time, most algorithms that have been proposed in literature are based on heuristics that require empirical assessment and comparison.

1.4 Objectives

1.4.1 General Objective

The general objective of this research is to compare the availability of virtual machine placement algorithms for cloud IaaS.

1.4.2 Specific Objectives

The following specific objectives will help to achieve the general objective of this research

- Understand state-of-the-art of cloud availability
- Identify metrics of availability
- Investigate VM placement algorithms and techniques
- Evaluate the availability of VM placement algorithms based on identified metrics

1.5 Methodology

In order to achieve both general and specific objectives of the research, different tools and techniques are used.

Literature Review

A systematic review of papers, articles and web sites will be conducted to have an understanding of cloud availability and VM placement as well as the challenges and prospects that are associated.

Experiment Environment

A well-defined environment for carrying out the empirical assessment of algorithms is vital as it is not feasible to conduct experiments in a real-world environment. The simulation tool that will be used for this experiment is CloudSim Plus version 3 which is an extension of CloudSim that has a mature and established simulation framework [10]. The tool has entities like PMs and VMs that can be extended and uses NetBeans integrated development environment (IDE) version 8.2 for developing with Java version 1.8.

Analysis

Generated data from simulation tool is statistically analyzed to obtain results which are as accurate as possible. Statistical tool IBM SPSS Statistics version 21 and Microsoft Office Excel 2016 are used to automate the analysis process.

1.6 Scope

This research is conducted on cloud IaaS availability which is one of the non-functional aspect besides performance, reliability and so on. Focus is on VM placement algorithms using metrics to measure availability. A single dimension is also considered in which Single-DC with all PMs having the same capacity to pack the VMs as one-dimensional objects into the minimal number of unit-capacity PM.

1.7 Contribution

The differences between the algorithms availability highlight the importance of empirical studies. Instead of comparing algorithms against performance and energy-saving only, considering availability along with other qualities would further foster the development of high-quality algorithms. This will also provide ethiotelecom with the selection of an optimal algorithm with respect to availability in the implementation of future cloud computing solutions although it is currently hardcoded in the cloud toolkits with a limited choice of preconfigured placement policies. Future versions of cloud computing solution will include a policy decision module to let providers like ethio telecom to specify their own resource selection policies and strategies

1.8 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 will discuss availability within the context of cloud computing as well as metrics and faults generation. It will also give a brief discussion on cloud virtualization and placement algorithms. Chapter 3 will present related works in the analysis of VM placement algorithms. Chapter 4 introduces the experimental setup along with selected algorithms. It will also describe the simulation tool and evaluation mechanism. Chapter 5 contains the experiment results followed by discussion. The last chapter, Chapter 6 will draw conclusions based on the findings and state future research directions.

Chapter 2 Literature Review

2.1 Cloud Availability

Availability is a non-functional requirement of cloud computing specified in terms of the percentage of time a system or a service is accessible. This percentage determines the allowed outage time for a given period [3]. Availability together with reliability and security are characteristics used to describe the time-dependent characteristics of a dependable system.

Four common classification of availability types have been proposed [30]. These are

- Instantaneous availability which is defined as the probability that a system is operating at any given time and contains information on maintainability. The condition requires a function that can be properly provided during time t with probability R(t) or provide the required function since the last repair time like u, 0 < u < t, with the following probability:

$$\int_0^t R(t-u)m(u)du$$

where m(u) is the system renewal density function. The availability is calculated as:

$$A(t) = R(t) + \int_0^t R(t-u)m(u)du$$

- Mean availability that can be defined as the proportion of mission time or time period that a system is available for use over a specific period of time such as (0, T) and is defined by:

$$A_m(T) = \frac{1}{T}\int_0^T A(t)dt$$

- Steady-state availability which can be determined by calculating the limit of the instantaneous availability as the time goes to infinity. The steady state, the time approximates to about four times the MTBF in instantaneous availability, can be calculated by using the following equation:

$$A(\infty) = \lim_{T\to\infty} A(T)$$

- Operational availability that evaluates the system availability including all the downtime sources, such as diagnostic and logistic downtime and is calculated as:

$$A_0 = \frac{\text{Uptime}}{\text{Operating cycle}}$$

where the operating cycle is the overall time of the investigated operation period whereas Uptime is the system's total functional time during the operating cycle. Based on this, ITIL (Information Technology Infrastructure Library) which is a framework that is adapted by many companies to manage and operate IT services, defines availability as ratio of elapsed time between service getting up and down to the total of elapsed time to repair a configuration item or IT service plus elapsed time between service getting up and down [32].

$$\text{Availability} = \text{MTBF}/(\text{MTBF}+\text{MTTR})$$

where MTBF is the mean time between two failures and MTTR is the mean time to repair. MTBF is the average time period between two consecutive component failures that require repair and high MTBF implies high availability. MTTR depicts the duration of the repair and recovery time which can be automatic or manual [12].

2.2 Cloud Availability Metrics and Faults

Cloud brings new facets to availability requiring a good basis for comparison. Features associated with availability can be reviewed using three dimensions: availability mechanisms, failures protected against, and metrics with further dimensioning into basic aspects and forms a structure that provides better insight into the different solutions.

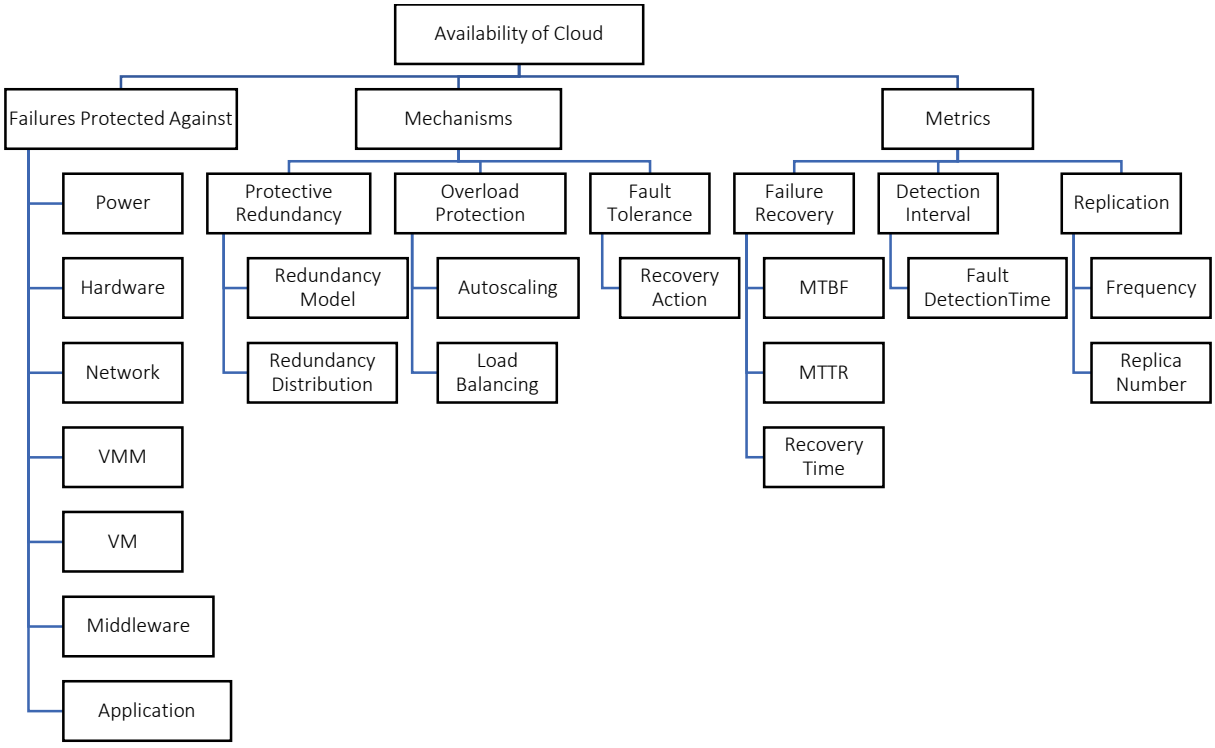

Figure 2.1 A taxonomy for availability in cloud computing [3]

The availability in the cloud requires a solution that can penetrate all the levels (SaaS, PaaS and IaaS) participating in the delivery of the cloud service as each of these levels have vulnerabilities to failures. Depending on a particular cloud solution, different vulnerabilities are either within or out of scope. For IaaS service model, the scope is related to guaranteeing protection against hardware and low-level software failures. The mechanisms to protect against failures can be categorized into three groups, namely fault tolerance mechanisms, protective redundancy, and overload protection which is displayed on a taxonomy shown in Figure 2.1. Fault tolerance is the capability of a system to continue normal operation even when a fault occurs. Protective redundancy is about the usage of redundant elements organized in different ways and collaborates following different rules depending on a specific redundancy model. Cloud service is a component-based distributed system with each component having different capacity limitation that may cause performance degradation or failures. Auto-scaling complemented by a load balancing provides protection against overload while optimizing resource utilization and distributing load among the computational resources. Metrics used by cloud providers to measure aspects associated with availability are grouped into three categories. Failure recovery includes MTBF, MTTR and RT (Recovery time) which is the mean

time to recover the service after a failure. Service recovery can be achieved by failing over the service to a redundant element and it does not necessarily imply the repair of the failed element. Detection interval and replication metrics are related to the maximum time interval necessary to detect a failure and the state synchronization mechanisms between the active and standby elements [2].

High expectations of cloud users are increasing the challenges that cloud providers are facing with respect to availability and fault tolerance. Due to system complexity of cloud, data centers are subject to some degree of failures and traditional approaches are less effective in addressing availability. Although researchers and companies are developing models, methodologies that use validation tools such as simulation are used as an easy, flexible and fast solution [13]. A methodology, proposed by Nita et al. in [14], uses a fault injector module for CloudSim tool to test and validate cloud infrastructure by inserting faults based on statistical distributions. They have tested and validated the statistical distribution influence on failures generated and observed CloudSim behavior with its current state and implementation to reproduce faults in a more natural and realistic way. Therefore, in order to compare VM placement algorithms, implementing fault generation module based on statistical distribution will have a big role in this research as it is not feasible to implement actual cloud infrastructure.

2.3 Virtualization in Cloud

Cloud computing provides three types of services namely SaaS, PaaS and IaaS. IaaS in contrast to SaaS and PaaS does not provide service application to customers. Instead, it offers computing infrastructure service like the delivery of operating system and virtualization technology. Virtualization is defined as partitioning resources such as processors, memory and input/output devices of a server into multiple segregated virtual machines with the goal of improving the sharing and utilization of the computer system. Virtualization allows data centers the movement of virtual machine in order to consolidate services on a smaller number of physical servers which involves virtual machine migration[15]. Three main characters of virtualization with respect to cloud computing are isolation, consolidation and migration. Isolation is done by confining program instructions inside VMs in order to achieve security and reliability. Consolidation is done by the

amalgamation of heterogeneous workloads onto a single physical platform for better system utilization. Migration facilitates hardware maintenance, load balancing, and disaster recovery by encapsulating a guest OS state within a virtual machine and allowing applications to be suspended, fully serialized, migrated to a different platform and resumed immediately or preserved to be restored at later date.

Cloud layers, as shown in Figure 2.2, consists of tasks that are executed on VMs in which it resides on a host (physical servers). The physical machines aggregate to form a datacenter entity that may be dispersed at different geographical locations. Hosts, with multiple cores for parallel processing, are responsible for assigning processing cores to the virtual machine. Tasks are executed on a virtual machine and must be mapped on to appropriate virtual machine based upon its configuration and availability. Virtual Machine Monitor (VMM) is a process that resides between the hardware and the host OS and must be installed on each host of the data center for taking care of decisions related to VM creation, destruction and scheduling.

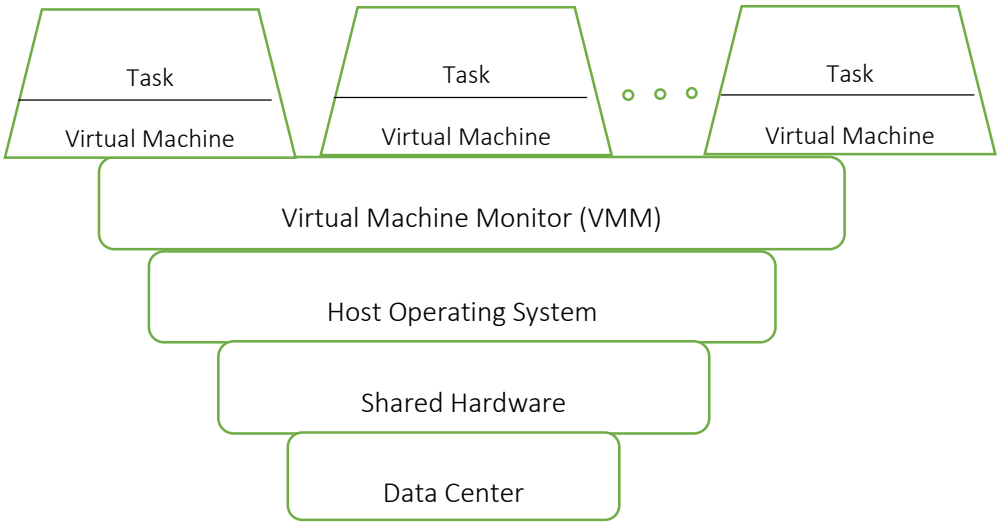

Figure 2.2 Cloud layers

The dynamic change in the resource requirement of a VM in a cloud environment can be addressed by either providing additional resources or by migrating the existing VMs from one host to another. VM migration is the process of moving VM from one physical machine to another physical machine

[8]. VM migration solves the problem of fault tolerance, load balancing, resource consolidation etc. Steps that are involved in the virtual machine migration process include:

- Identifying overloaded or underloaded physical machine
- Selecting virtual machines
- Selecting the physical machine on which the selected virtual machines are to be placed by avoiding migration increase, resource wastage and energy consumption
- Moving the virtual machines

Virtual machine placement, part of the virtual machine migration, is the process of selecting the appropriate host for the given virtual machine when a request for the VM comes to the VMM.

Virtual machine placement reduces the cost for cloud providers and increase the return on investment (ROI) by achieving goals such as

- Reducing inter-data center traffic in federated cloud and intra-data center traffic by decreasing VM to VM and VM to storage(data) traffic
- Mitigating energy consumption by minimizing the number of active servers and networking elements
- Increasing security
- Maximizing resource utilization: The effective and efficient utilization of each server reduces the need for more PMs to host VMs.
- Minimizing SLA violation by maintaining high reliability, performance and availability using load balancing and replication

If the number of physical and virtual machines are less, static VM placement methods can be applied, but if the number of virtual machines and physical machine is more dynamic, VM placement methods are required.

Mathematically, let n be the total number of virtual machine and m be the total number of host. Then, the number of possible mapping can be calculated by the following equation [16]

Number of possible mappings = m^n , which is an NP hard problem.

VM placement algorithms can be classified based on approaches into Power-based and QoS-based. Power-based approach is to save the energy in such a way that each server can be utilized at its maximum efficiency while the QoS based approach focus is on maximizing the QoS delivered by the service provider with respect to VM-PM mapping. The approaches can be further classified based on the necessity of migration which determines the type of techniques to be used. When migration is required, power-based and QoS approaches can be implemented with deterministic, heuristics and meta-heuristics. When migration is not required, QoS approach with deterministic, heuristics, meta-heuristics and other power expand min-max algorithm techniques are used while there is no technique for the power-based approach [5].

Classical deterministic techniques include Constraint Programming (CP), Linear Programming (LP), Integer Linear Programming (ILP), Mixed Integer Linear Programming (MILP), Pseudo-Boolean Optimization (PBO) and Dynamic Programming (DP) that propose mathematical formalizations of the VMP problem. Constraint programming, for instance, is a programming paradigm used in the combinatorial search problems where some constraints are applied and must be fulfilled in relations between variables [17].

Bin packing algorithms use heuristics that include algorithms such as Fist-Fit, First-Fit-Decreasing, Best-Fit, Best-Fit-Decreasing, Worst-Fit and Heaviest-Fit to accomplish results[18]. It is a problem where objects of different volumes must be packed into a finite number of bins in such a way that the number of bins used is minimized. The VM placement problem can be designed using bin packing problem by placing as many VMs into a single PM so that the number of PMs required to pack the VMs is minimized.

Meta-heuristics which are useful to obtain solutions in practical time include Memetic Algorithms (MA), Particle Swarm Optimization (PSO), Stochastic Integer Programming(SIP), Ant Colony Optimization (ACO), Genetic Algorithm (GA), Neighborhood Search (NS), Cut-and-Search, Simulated Annealing (SA) and Tabu Search (TS) [13].

Stochastic integer programming, for example, is used for optimizing problems with uncertainty that include unknown parameters and cannot be captured by deterministic integer programming to find the suitable host which consume less energy and minimize the resources wastage.

A genetic algorithm, also categorized as global search heuristics, is a search technique used to find approximate solutions to optimization and search problems. It requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. The VM placement problem goal would be to deliver a solution that is nearly optimal in terms of the number of bins used and the efficiency of packing of the bins.

Simulated annealing algorithm is also for optimization problems that cannot be managed using combinatory methods as the number of objects becomes large to find a very good solution even in the presence of noisy data. Approximation algorithms, such as p-approximation, provide measurable solutions unlike heuristics and meta-heuristics where the value of a solution will not be more (or less) than a factor p times the optimum solution.

In short, deterministic like constraint Programming are useful for cases where the demands of the VMs are known beforehand and the number of constraints determines the time to generate a solution. Heuristics like, bin-packing is useful for dynamic VM placement where the demand is highly variable and will always generate a good solution in a considerable amount of time with physical machines having the same memory and CPU. Stochastic integer programming is useful if probability distributions can be computed for uncertain parameters. Genetic Algorithm requires more computing time and higher computing resources as compared to bin packing and useful for static placements, where the demands do not vary over a considerable period of time[19]. Simulated Annealing Algorithm is very suitable in static VM placement problem.

The Cloud Ecosystem management tools for data centers such as the Platform VM Orchestrator, VMware vSphere, and Ovirt, meet resource management requirements on a pool of physical resources providing features such as dynamic placement and server consolidation. Although the resource selection algorithm is currently hardcoded in the cloud toolkits with a limited choice of preconfigured placement policies, future versions will include a policy decision module to let providers specify their own resource selection policies and strategies [15]. Strategies followed by Open-Source Solutions like Nimbus, Open Nebula and Eucalyptus decide allocation by using rank, greedy and round-robin algorithms. Open Nebula algorithm takes requirements and predefined rank as its input and produces the resource number in which to place the virtual machine as output by sweeping away the resources which are not befitting into the requirement. For placement of a virtual

machine on the underlying host Eucalyptus and Nimbus uses greedy strategy so that whichever node that can run the virtual machine found first is selected as the host for virtual machine placement. For placement of a virtual machine on the underlying host Eucalyptus and Nimbus also use Round-Robin algorithm by recording the last position of the visited where resources are thought of as a circular linked list[20].

2.4 Heuristic Based VM Placement

Bin packing approach has numerous simple and primary heuristics-based VM placement algorithms and some are listed as follows,

Firs-Fit

First-Fit scheduler uses a greedy approach to allocate the first free partition large enough that can accommodate the VM and finishes after finding the first suitable free partition. When each VM is then taken off the list and an attempt is made to place it on the first PM, the sum of resource demands of all VMs on that PM is checked against the total capacity of the PM. If none of the physical machines satisfied the resource requirement of the VM, then new PM is activated and assigns VM to the newly activated PM. The main advantage is that it searches as little as possible although resource imbalance is a possibility due to remaining unused resources left after allocation [21].

First-Fit-Decreasing

First-Fit-Decreasing, a variant of the First-Fit-Algorithm, can further reduce the average number of active hosts by sorting both physical machines and virtual machines from high capacity to least capacity after which first fit process takes place. The main advantage is that the virtual machines in the queue are ordered according to some criteria before placement. The asymptotic running time of FFD is O (nlogn+ nm), where n is the total number of VMs and m denotes the total number of PMs. O(nlogn) is the additional cost of running time for sorting algorithm [21].

Best-Fit

When VM arrives, scheduler deals with allocating the smallest free partition which meets the requirement of the requesting process by visiting the decreasing order of the physical machines capacity used in and place the VM to the first PM that has the enough resources. Memory utilization

is much better than First-Fit as it searches the smallest free partition first available. Problem with this approach is that it can increase the resource unbalancing by filling up memory with tiny useless holes which makes the process even slower.

Best-Fit-Decreasing

The Best-Fit Decreasing algorithm first sorts the VMs in the decreasing order of its VM utilization. If each host has enough resource for the VM, then the scheduler selects the host as a destination, otherwise, it does nothing. The Best-Fit algorithm keeps a set of hosts active at all times and places each VM at the best position to achieve load balance among the active hosts that would increase the probability of success in VMPs. The asymptotic running time of BFD is same as FFD [21].

Worst-Fit

In the Worst-Fit algorithm, the scheduler places the VMs to less active servers by sorting PMs and VMs in decreasing order of utilization. Then, the first PM which has the required resources from the list of the sorted PMs is selected and then deployed. This procedure is repeated until all the VMs are mapped to the PMs. The advantage is reduction in the rate of production of small gaps as it is the reverse of Best-Fit. For VMs requiring larger resource that may arrive at a later stage, availability problem arises as it cannot be accommodated as the largest resource is already split and occupied [21].

Next-Fit

Next-Fit is a modified version of First-Fit algorithm that uses a variable called next which initially is null. Placement method considers the first virtual machine first node where the search starts from. If the physical machine satisfies the resource requirement, the virtual machine starts on that physical machine and next is replaced by the current physical machine. If there is no PM with sufficient available resources, a new PM will start up and create the requested VM on the newly started PM. For the next virtual machine, the search starts from next to the stored PM and search continues.

Random-Fit

In Random Fit, the scheduler starts searching in a random manner and if the physical machine satisfies the resource requirement, the virtual machine starts on the physical machine. If it is not

satisfied randomly, another physical machine is chosen. This process continues until a PM is found that satisfies the requirement of the virtual machine[22].

Round-Robin

In Round-Robin, the scheduler which finds the next PM having suitable resources to place a given VM in a circular way by moving to the next suitable PM when a new VM has to be placed. It does this by recording the last position of the scheduler visited starting from the last visited position next time a new request comes taking into consideration that resources are in a circular linked list [15].

In this thesis work, the focus is on a family of packing heuristics First-Fit, Best-Fit, Worst-Fit and Round-Robin motivated by algorithms for bin packing that can be meaningfully compared to each other and time constraint for the thesis work.

2.5 Chapter Summary

This chapter gives emphasis on defining cloud availability and its common classifications along with faults and metric taxonomy based on different reviewed kinds of literature. Main characters of virtualization are also reviewed in order to show the importance of VM placement algorithms which are responsible for assigning the VMs to a specific PMs. Finally, selected bin packing based simple heuristics VM placement algorithms are investigated in order to make a selection that will be used for comparison.

Chapter 3 Related Works

In this section, six papers related to cloud availability and VM placement algorithm analysis are reviewed. Mann et al. [23] stated that heuristics algorithms which have been proposed previously have no theoretical guarantee on their effectiveness. They proposed that empirical assessment and comparison of heuristics algorithms should play an important role. In their paper, an environment for experimentally evaluating and comparing the performance of VM placement algorithms was presented. It was built on the popular open-source CloudSim toolkit and extended with input data format, converters for publicly available workload traces, and workload generation facilities. The evaluation was performed on the comparison of seven algorithms that solve the same version of the VM placement problem. What their findings showed was that algorithms which perform well on one metric perform poorly on the other metrics and energy consumptions can be as much higher than the expected along with the heterogeneity of the PMs with respect to both capacity and power efficiency. Between the evaluated algorithms, two algorithms gave the best results in terms of power and performance efficiency.

In [24], Shi et al. address the problem of vector bin packing strategy to use in an attempt to consolidate VMs in a data center and reduce power consumption. In the paper, they showed that by calculating the CPU and memory components of each and then sorting the VMs and PMs, a significant energy saving could be achieved given the initial placement of the VMs and PMs. They proved that although ILP formulation yields an optimal solution, it is not suitable for the dynamic runtime placement of newly arrived VM requests. Therefore, even if sub-optimal, FFD approaches to compute the placement are required for the provisioning of VMs as the method is common for solving bin packing problem for both online and offline scenarios. In their work, six bin packing algorithms based on FFD using six different sorting strategies were proposed. The placement was created with a real cloud scenario that randomly generated VMs from the VM types that are placed on PMs in a random mapping to recreate the dispersed state of a data center. Metrics also measured various facets of the algorithm performance such as the number of migrations, the number of PMs used and PM utilization percentage. For each test case, they started with a data center with initial placement from time 0, and the number of VM requests modelled as a Poisson process with an average $\lambda = 20$. They evaluated a number of bin packing algorithms using different sorting strategies

to determine which one is most effective at consolidating VMs in the data center enabling unused PMs to be turned off. Furthermore, various strategies were developed to pack dynamic VM requests in the most efficient way. They showed that six of the algorithms offered some improvement in the number of PMs which could be powered off due to consolidation, even when the number of VM migrations needed for this consolidation was also taken into account with an overall one best performing algorithm.

In the IaaS cloud availability analysis of Bo et al.[25], the emphasis was given to explore stochastic modelling and sensitivity analysis techniques for analyzing the impact of repair policy and system parameters on the IaaS availability by examining repair policies. The numerical analysis results showed that different IaaS policy maintains a different level of availability based on cost as long as there are enough repair facilities and small MTTR. Sensitivity analysis was used to allow the exposure of system QoS.

Aaron et al. [26] described a general methodology that can be used to measure the availability of arbitrary computer systems. The methodology uses fault injection to provoke situations where availability may be compromised and data to be generated. They showed that availability is measured in terms of quality of service by establishing a standard definition and metrics that can be used to report the results to be analyzed. They were able to map transient faults into failure conditions and quantify the quality of service and availability delivered.

A comprehensive and empirical performance-cost analysis of provisioning and allocation in IaaS clouds was analyzed by Villegas et al. [27]. They introduced a taxonomy based on the type of information used in the decision process and mapped it to eight provisioning and four allocation policies. Then, analyzed the performance and cost of policies through experimentation. They showed that policies that dynamically provision and/or allocate resources can achieve better performance and cost. They also looked at the interplay between provisioning and allocation and show preliminary results. They have developed SkyMark, a framework for IaaS performance evaluation, and conducted with it empirical research in three IaaS clouds, including Amazon EC2. Due to actual resource and budget. They concluded that none of the tested (combined) policies is consistently better than the others across all cases but static provisioning policies deliver better performance when there are enough resources and incur up to 5 times higher cost. Also, allocation

policies with information about job run times achieve significantly better performance than uninformed allocation policies, when coupled with dynamic provisioning policies.

Chowdhury et al. have implemented some of the bin packing solutions to address VM placement problems trying to put the focus on minimizing power consumption without making drastic alterations over the other areas, to meet the quality of IaaS in their work [21]. They ran the simulation on a cloud computing simulation toolkit CloudSim using Planet Lab workload data. In their work, they followed heuristics for dynamic consolidation of VMs based on the previous data of resource usage to design their proposed VM placement algorithm and were successful to make remarkable improvements over the existing solution that managed to get lower power consumption, less amount of SLA violation and performance degradation over the existing VM placement algorithm. They found out that local regression-based algorithm equipped with minimum migration time VM selection policy significantly outperforms other dynamic VM consolidation algorithms if power consumption, SLA violation, performance degradation due to VM migration, SLA violation time per active host and number of VM migration taken into account.

In summary, the reviewed works of literature provided an understanding in the area of cloud availability comparison for VM placement algorithms in terms of parameters, design and methodologies. However, empirically they were very limited in the coverage of algorithms with respect to availability and most of them had a different focus from this thesis work.

Chapter 4 Experiment Setup

For carrying out the empirical assessment of algorithms, reproducible experiments must be conducted in a real-world environment to reach conclusive results. But, due to the unavailability of such environment, simulation is used as the basis of the assessment. A simulation represents an environment in which a system that behaves similarly to another system, but is implemented in a different way by providing the basic behavior of a system. It may not reproduce the exact output as the real one unlike emulation which presents a system that behaves exactly like another system with the same output. For this research, evaluation simulator known as CloudSim Plus is used which is a fork of CloudSim written in Java language [10]. CloudSim and CloudSim Plus are almost similar with the exception of re-engineering done on CloudSim Plus to remove code duplication and compliance to software engineering standards. Environment setup, as shown in Figure 4.1, uses CloudSim Plus as the central simulation engine and extended with a facility to inject fault to the cloud. VM placement algorithms can then be compared by means of defined evaluation metrics.

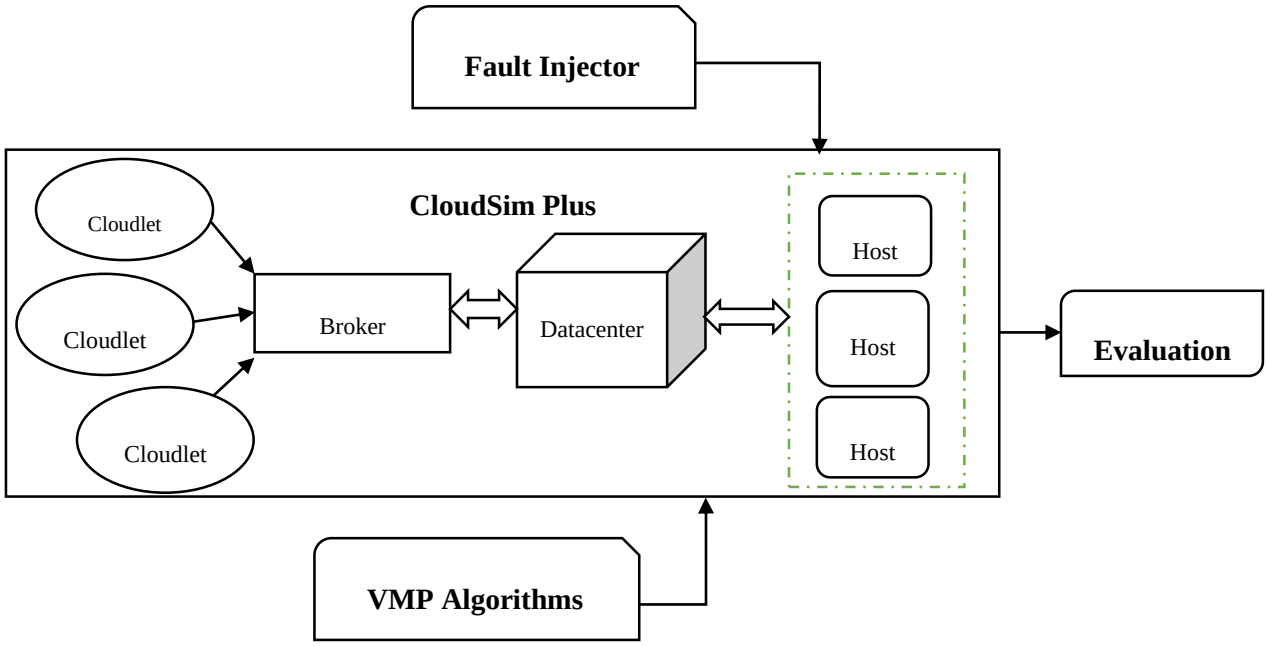

Figure 4.1 Environment Setup Overview

4.1 CloudSim Plus

Computer simulation enables researchers to conduct experiments without the cost of cloud services and also provide a faster way to run experiments in just a few minutes or seconds that would take hours or days to run in real infrastructure. CloudSim, a generalized open source and extensible simulation framework for cloud computing, is developed in Java providing great flexibility to create simulation scenarios. The shortcomings were limited documentation, lack of a more organized package structure and duplication of code that made maintainability, extensibility and testing difficult to manage. CloudSim Plus, based on CloudSim 3, went through an extensive redesign and re-engineering process to provide an updated, modern, highly extensible and easier-to-use cloud simulation framework. It is a Java based simulation framework that allows modelling and simulation of different cloud computing services, ranging from IaaS to SaaS layers [10]. Implementation of scenarios for assessment and validation of algorithms using the framework allows researchers and developers to specify the characteristics of different entities of a cloud provider. Entities such as physical machines, storage area networks and VMs enable virtualizing physical and logical resources based on requirements and behaviors of applications and workloads. By extending the basic framework and implementing algorithms, it is possible to achieve goals such as load balancing, energy-saving, fault-tolerance, scalability, elasticity, SLA, network traffic, communication delay and so on.

CloudSim Plus has a simpler module and package structures that are compounded into four modules re-organized and inherited from the parent project. Figure 4.2 presents the current project architecture with testbeds and benchmark modules that are new in CloudSim Plus.

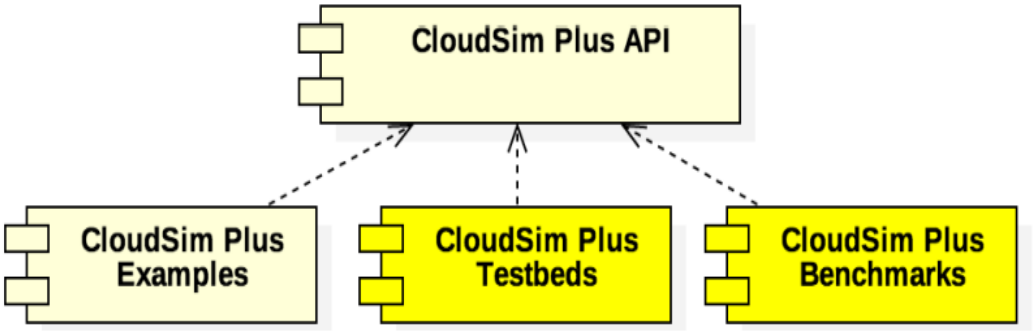

Figure 4.2 CloudSim Plus Architecture [10]

CloudSim Plus API is the main independent module that contains the framework API to build simulation scenarios. This module can be added as a Maven dependency when building simulation scenarios. CloudSim Plus Examples provide CloudSim examples that are refactored and well organized to be used in CloudSim Plus API along with new examples for CloudSim Plus. CloudSim Plus Testbeds implements some simulation testbeds by providing base classes that allow a researcher to collect valid scientific results with examples on how to create broader testbed experiments. CloudSim Plus Benchmarks is used internally to measure the overhead of CloudSim Plus features. CloudSim Plus simulation framework has also introduced exclusive features such as on-demand creation of VMs and cloudlets, vertical and horizontal VM scaling, parallel simulation execution, event listening, and implementation of classes for heuristics.

CloudSim Plus main entities are:

- Cloudlet represents the task for the cloud characterized by length and number of PEs
- Broker mediates between cloudlets and datacenter by monitoring cloudlets status and requirements
- Datacenter manages available resources like hosts, PEs, VMs, and memory
- Virtual machine represents a software-based emulation of a host
- Host represents the physical resource characterized by a number of PE, CPU and RAM
- Processing element (PE) represents a unit of the system responsible for the completion of a task

The configuration for the availability evaluation of VM placement algorithms using CloudSim Plus simulator with a single data center is shown in Table 4.1.

Host	**No. of Physical Machines**	**10**
	RAM	500GB
	No. of PE	4
	MIPS/PE	1000
VM	No. of VMs	4
	RAM	10GB
	No. of PE	2
Cloudlet (Tasks)	No. Cloudlets	10
	Instruction Length	2800000000
	File size	300
	No. of PE	2

Table 4.1 CloudSim Plus Datacenter Configuration

4.2 Fault Injection

Cloud failures can be broadly classified into software faults, hardware faults and network faults [28]. Software faults are related to infrastructure service failure while hardware faults are related to device or machine incidents such as virtual or physical node outage. And network faults are concerned with the network infrastructure such as routers, switches and so on. This research focus is on statistically distributed hardware failures that are injected to impact the availability of IaaS service with respect to VM placement algorithms. That is, at random moments of time, fault injector module will generate an event that will simulate a failure in the host of the datacenter based on discrete statistical distribution.

Various statistical distributions exist in CloudSim Plus such as Exponential, Uniform, Gamma, Lognormal, Lomax, Pareto, Poisson, and Weibull. From these distributions, Poisson is used in experiments where random events are following the pattern of a discrete probability distribution that can be used to calculate the probability of certain event number to occur in a fixed interval of time and space. The Poisson distribution probability function for a given discrete random variable of independent events with a known average occurrence rate has the following definition:

$$f(k;\lambda) = \Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where X = k means the actual number of success resulted from the Poisson experiment, λ is the average number of successes that occurs in a certain known interval [14]. Each entity of CloudSim Plus can send a certain event to another. The broker sends cloudlets to the datacenter that will schedule it according to scheduling policy on a host. As shown in Figure 4.1, the fault injection module for the simulation will generate a failure in the cloud system with an average number of failures expected to happen each hour in a Poisson Process which is also called event rate. Based on the assumption that one failure occurs every hundred hours, i.e., λ =1/100=0.01, the fault injector will randomly inject failures according to the Poisson probability distribution to a random PE of a host.

4.3 Algorithm Selection

The VM placement algorithms to be compared for availability are implemented in CloudSim Plus by overriding VmAllocationPolicyAbstract class. The focus of this research is on a family of packing heuristics inspired by algorithms for bin packing that can be meaningfully compared to each other and within the time frame of this thesis work. From the list of algorithms described in Section 2.4, First Fit, Best Fit, Round-Robin and Worst Fit are selected for comparison. These algorithms differ mainly in the order in which the VMs are considered and the order in which the PMs are considered for a given VM. First Fit, Best Fit and Worst Fit search start sequential from the first PM for VM placement while Round Robin starts from the last PM selected in the previous placement. The asymptotic running time of First Fit and Best Fit is O(nm) and $O(nm^2)$ while Round Robin is O(nm) where n denotes the total number of VMs and m denotes the total number of PMs [20]. The

"lowest common denominator" versions of the problem are selected where a single datacenter with hosts having the same CPU size and memory capacity.

4.4 Evaluation

After setting up CloudSim Plus on a windows machine with a dual-core processor, configurations of hosts, VMs and cloudlets were made on the default XEN hypervisor as shown in Table 4.1. The number of hosts, VMs and cloudlets were made based on a number of experiments on the simulator with a Poisson process fault arrival rate of $\lambda = 0.01$. The last step in the experimentation process is to collect data from the simulation runs and evaluate availability of the four algorithms based on metrics. The selected metrics for this research, MTTR and MTBF, were used to evaluate the availability of each algorithm along with CPU and memory utilization. The availability comparison for each of the four algorithms was made by running twenty simulations and injecting fault to the PEs of a host based on a Poisson distribution. Each availability calculation was based on the MTTR and MTBF metrics that the simulator measured. A 10%, 50% and 90% utilization of CPU and memory were also considered for comparing the availability of the algorithms. Using the extended CloudSim Plus classes, all measurements were generated and post-process statistical analysis was done using a standard spreadsheet program Mircosoft Office Excel and a statistical tool SPSS. In the analysis, the above metrics were used to characterize the availability of the algorithms to obtain a clear picture about the strengths and weaknesses of each algorithm which lead to the assumption that different algorithms realize different trade-offs based on metrics.

4.5 Chapter Summary

This Chapter mainly focused on the experimental setup that is used for the comparison of VM placement algorithms. It describes the simulation tool that is used along with the fault injection methodology to inject fault at a random moment of time. In addition, algorithms were selected that are to be compared to each other based on the family of packing heuristics. Finally, the evaluation process is elucidated based on the selected configuration.

Chapter 5 Results and Discussion

5.1 Results

In this research, statistical univariate analysis has been used to analyze the availability data. The availability variable takes a value between zero and hundred and uses ratio scale of measurement. Both descriptive (for describing the central tendency, dispersion and distribution) and inferential statistics (for making an inference like hypothesis testing) are used in the analysis of availability.

5.1.1 Descriptive Statistics for Algorithm Comparisons

As the name implies, descriptive statistics are used to describe and summarize data in a meaningful way so that some pattern may be gathered from it.

5.1.1.1 Based on Availability

Statistics	Worst Fit	First Fit	Best Fit	Round Robin
Mean	91.22	79.55	80.87	82.86
Median	99.26	82.12	82.87	89.13
Mode	100.00	100.00	100.00	100.00
Std. Deviation	13.23	18.54	17.18	19.09
Variance	175.06	343.83	295.05	364.39
Skewness	-1.58	-0.83	-0.57	-0.71
Kurtosis	1.65	0.37	-0.90	-0.99

Table 5.1 Descriptive Statistics of Compared Algorithms

Examining the statistics of the four algorithms using central tendency, Table 5.1 reveals that the mean of Worst-Fit algorithm has the highest availability while First-Fit is the lowest. This leads to the notion that Worst-Fit has high availability. But the skewness and kurtosis of the four algorithms as shown in Figures 5.1-5.4 is different from zero. This means the distribution does not follow a normal distribution and the mean simply doesn’t provide the accuracy needed for the decision.

Measuring the dispersion, standard deviation shows that the spread of the data from the mean appears to be low in all the algorithms since the Coefficient of Variation(CV) is less than one although the measure of data points or variance from the mean is high specifically for Round-Robin algorithms.

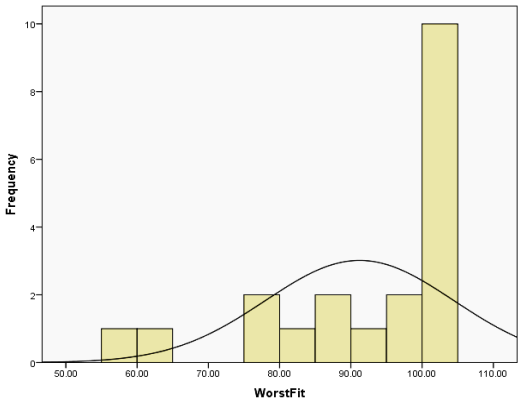

Figure 5.1 Distribution of Worst Fit Algorithm

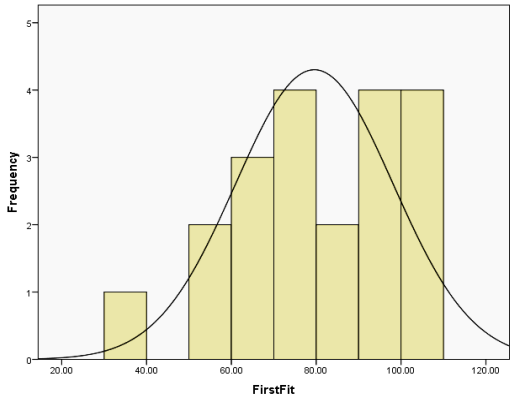

Figure 5.2 Distribution of First Fit Algorithm

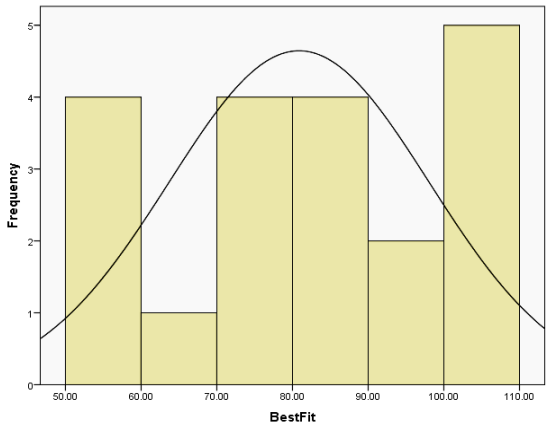

Figure 5.3 Distribution of Best Fit Algorithm

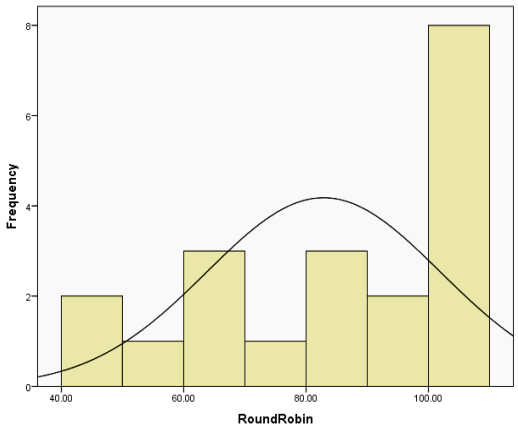

Figure 5.4 Distribution of Round Robin Algorithm

5.1.1.2 Based on CPU utilization

CPU utilization data was generated at three levels of utilization 10%, 50% and 90% by setting a full memory utilization. CPU utilization statistics of the four algorithms with central tendency on Table

5.2 reveals that the mean of Round-Robin algorithm availability is the highest on 10% while Worst-Fit is highest on 50% and 90% utilization levels. But the skewness and kurtosis of the four algorithms at the three utilization levels is different from zero making the distribution not to adhere to a normal distribution. Measuring the dispersion, standard deviation appears to be low in all the algorithms since the CV is less than one although the measure of data points or variance from the mean is high specifically for Round-Robin algorithms on 10% utilization level.

	10% Utilization				50% Utilization				90% Utilization			
Statistics	**Worst Fit**	**First Fit**	**Best Fit**	**Round Robin**	**Worst Fit**	**First Fit**	**Best Fit**	**Round Robin**	**Worst Fit**	**First Fit**	**Best Fit**	**Round Robin**
Mean	35.10	19.20	19.45	37.10	84.39	73.81	72.09	82.70	83.82	70.78	64.44	82.67
Median	35.00	17.50	21.00	27.50	99.29	81.33	71.90	81.82	92.92	80.12	60.92	88.49
Mode	40.00	10.00	25.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Std. Dev.	17.15	9.38	5.48	29.84	20.69	22.36	20.33	16.61	18.18	27.25	24.93	19.82
Variance	293.99	87.96	30.05	890.20	428.15	499.90	413.45	275.90	330.43	742.77	621.49	392.99
Skewness	0.90	0.62	-0.45	1.42	-0.82	-0.20	0.26	-0.57	-0.95	-0.35	0.16	-1.43
Kurtosis	1.08	-1.58	-0.28	0.98	-1.17	-1.51	-1.41	-0.44	-0.32	-1.58	-1.20	2.14

* Multiple modes exist. The smallest value is shown

Table 5.2 Availability of Compared Algorithms at 10%, 50% and 90% CPU Utilization

5.1.1.3 Based on Memory Utilization

Memory utilization data was generated at three levels of utilization by setting full CPU utilization. Memory utilization statistics of the four algorithms with central tendency on Table 5.3 reveals that the mean of Worst-Fit algorithm availability is the highest on 10% and 90% utilization levels while Round-Robin is the highest on 50% utilization level. Skewness and kurtosis of the four algorithms at the three utilization levels is different from zero like the CPU utilization. Measuring the dispersion, standard deviation appears to be low in all the algorithms with CV less than one and variance from the mean is high specifically for Best Fit algorithms on 50% utilization level.

Statistics	10% Utilization				50% Utilization				90% Utilization			
	Worst Fit	First Fit	Best Fit	Round Robin	Worst Fit	First Fit	Best Fit	Round Robin	Worst Fit	First Fit	Best Fit	Round Robin
Mean	84.78	72.33	76.93	81.94	83.54	77.49	73.45	85.70	87.28	68.89	76.37	79.54
Median	92.46	69.78	75.74	94.43	90.36	72.89	78.78	90.23	96.64	64.06	84.93	78.61
Mode	100.00	100.00	41.80*	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Std. Dev.	18.32	22.15	17.24	22.55	18.75	17.44	25.86	15.53	16.63	25.29	24.83	16.69
Variance	335.45	490.81	297.05	508.51	351.59	304.18	668.65	241.12	276.50	639.77	616.49	278.46
Skewness	-1.14	-0.61	-0.31	-0.97	-1.07	0.05	-0.82	-0.48	-1.08	-0.21	-0.72	-0.39
Kurtosis	-0.02	0.27	-0.94	-0.45	0.37	-1.21	-0.39	-1.45	0.04	-0.93	-1.00	-0.45

* Multiple modes exist. The smallest value is shown

Table 5.3 Availability of Compared Algorithms at 10%, 50% and 90% Memory Utilization

5.1.2 Inferential Statistics for Algorithm Comparisons

Analysis is made for inferential statistics using the Kruskal-Wallis H test which is a rank-based nonparametric test that can be used to determine if there are statistically significant differences between two or more groups of an independent variable. It is a nonparametric equivalent to the parametric one-way ANOVA, and an extension of the Mann-Whitney U test to allow the comparison of more than two independent groups[31]. The first part of the process in using this test includes checking assumptions such as the level of measured dependent variable (ordinal or continuous), two or more categorical independent groups for the independent variable, independence of observations between observations in each group or between the groups themselves and similarities in the shape of distributions in each group that will determine the comparison in which the dependent variable will be based on (medians or mean ranks). The first three assumptions are fully met as the dependent variable is measured on a continuous level with four groups of independent variables that have independent observations both between groups and within each groups. The fourth assumption on the shape of the distributions, as shown in Section 5.1.1 Descriptive Statistics, makes it compulsory in the use of mean ranks for comparison due to the variation in skewness.

The Kruskal-Wallis H hypothesis test for availability will be take the medians for checking the statistical significance on the availability of the algorithms by comparing mean ranks among the four groups with null and alternative hypotheses as follows:

Null Hypothesis H_0: Availability medians are equal at 95% CL

Alternative Hypothesis H_A: Availability medians are not all equal at 95% CL

5.1.2.1 Availability K-W test

A Kruskal-Wallis H test showed that there was no statistically significant difference in availability between algorithms with full memory and CPU usages. The test statistics showed that Kruskal-Wallis H test value of 5.823 is attained with a statistical significance of 0.121 which is greater than the α 0.05 and degree of freedom 3. Using the mean ranks, the four algorithms were compared for availability making Worst-Fit attain the highest availability with 50.33 mean rank while First-Fit with lower availability of 34.45 as shown in Figure 5.5.

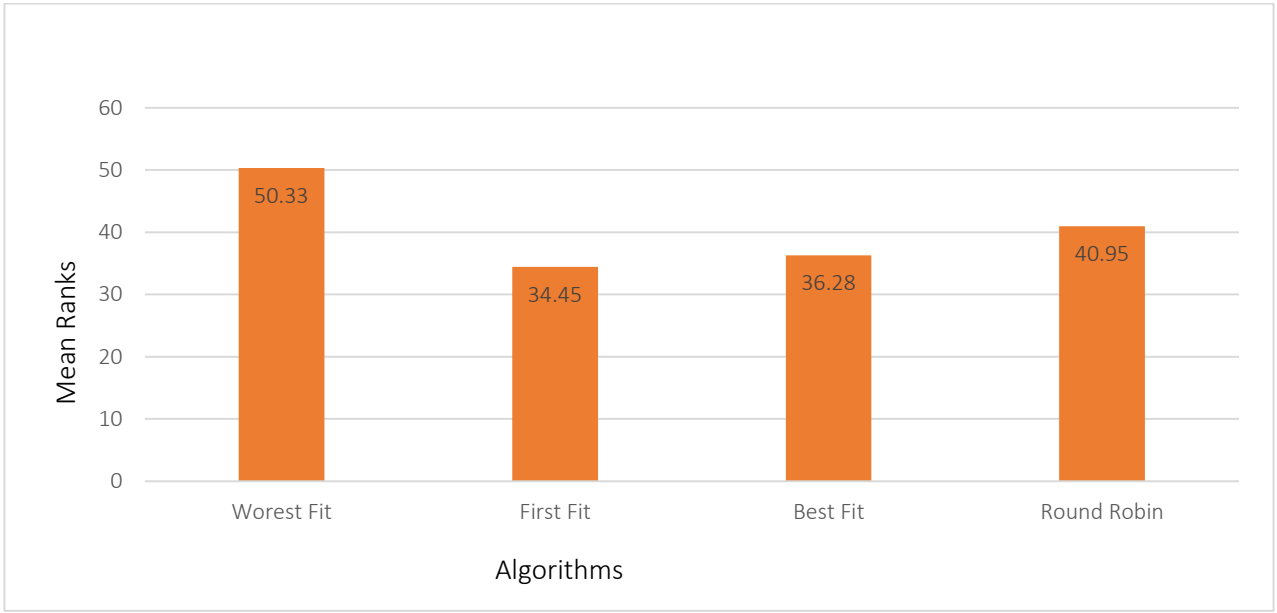

Figure 5.5 Algorithms Comparison Based on Availability

5.1.2.2 CPU Utilization Availability K-W test

A Kruskal-Wallis H test showed that there was no statistically significant difference in the availability of CPU utilization levels between algorithms with full memory usage. The test statistics showed that Kruskal-Wallis H test value of 6.303 is attained with a statistical significance of 0.0981 which is greater than α value 0.05 and degree of freedom 3. Using the mean ranks, the four algorithms were compared for availability making Worst-Fit the highest availability mean rank in the three CPU usage levels as shown in Figure 5.6.

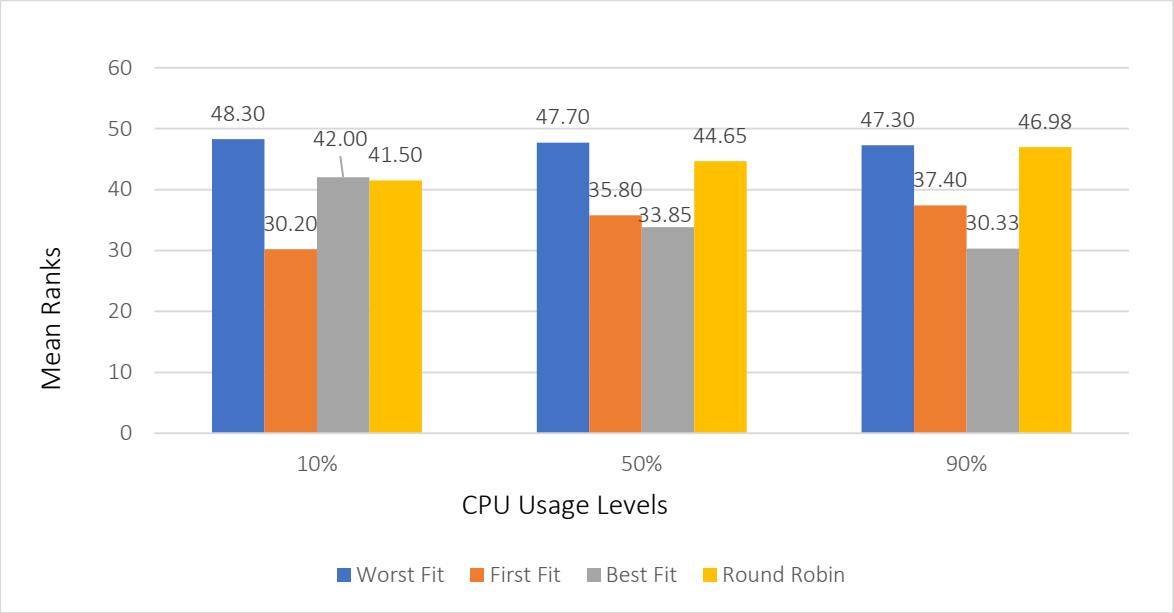

Figure 5.6 Availability based on CPU Loads with Full Memory Utilization

5.1.2.3 Memory Utilization Availability K-W test

A Kruskal-Wallis H test showed that there was no statistically significant difference in the availability of memory utilization levels between algorithms with full CPU usage. The test statistics showed that Kruskal-Wallis H test value of 5.982 is attained with a statistical significance of 0.112 which is greater than the α value 0.05 and degree of freedom 3. Using the mean ranks, the four

algorithms were compared for availability making Worst-Fit the highest availability mean rank at 10% and 90% usage levels while Round Robin at 50% usage level as shown in Figure 5.7.

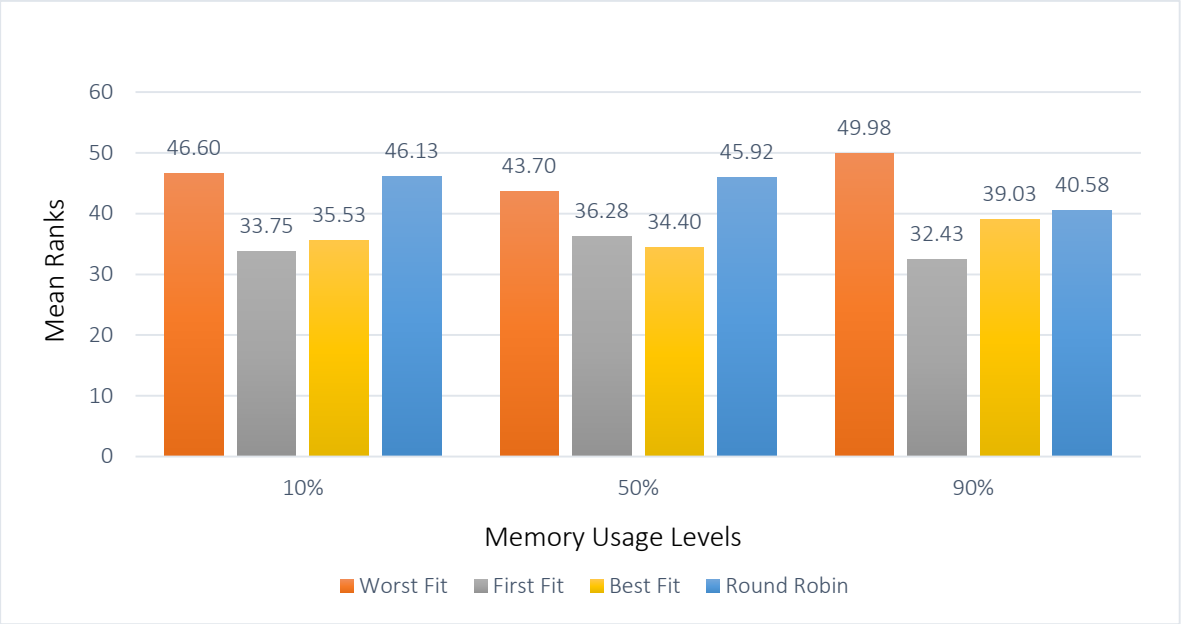

Figure 5.7 Availability based on Memory Loads with Full CPU Utilization

5.2 Discussion

Based on K-W test on the mean ranks with CL of 95%, the four algorithms were compared for availability and 15.88 mean rank difference exits between the highest executing algorithm Worst-Fit and least executing algorithm First-Fit. These gap between algorithms is not evident between the remaining three algorithms. One reason can be the placement of VMs to less active servers that is performed in Worst-Fit algorithm unlike the remaining ones.

In terms of CPU load utilization test at 95% CL, the availability gap shows a decrease between the highest and lowest executing algorithms as the CPU utilization increases from 10% to 50% and increases back when the usage level increases to 90%. The 10% CPU utilization gives Best-Fit a slightly better availability than Round-Robin although it is lower than Worst-Fit algorithm. In both the 50% and 90% CPU utilization, First-Fit algorithm has a better availability than its close rival Best-Fit algorithm by taking advantage of the decrease in searching unlike the 10% CPU usage level.

With respect to the three memory utilization levels tests at 95% CL, the availability at 10% and 90% utilization is highest for Worst-Fit algorithm while Round-Robin has the highest for 50%. Although their gap is insignificant when the utilization is low, it will increase when the usage goes to 90%. The availability gap between Best-Fit and First-Fit increases as the usage level increases from 10% to 90% although First-Fit has better availability at 50% utilization level. This may be due to the increased memory utilization of Best-Fit since it searches the smallest free partition first available.

The above findings show that the availability of IaaS in the cloud depends on VM placement algorithms that are influenced with the different levels of utilization of the physical resources. Thus, the selection of specific algorithm suitable for a particular configuration plays an important role in maintaining the maximum availability.

With respect to the limitation of the experiment, due to the sparse availability of real-world fault load data and time constraint, only statistically generated load is used for analysis. This gives the advantage of generating as much data as needed but some real-world failure patterns and frequencies may be missed subsequently swaying the availability comparison of VM placement algorithms.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This thesis work used a CloudSim Plus simulation environment which is publicly available for experimentally evaluating and comparing the availability of VM placement algorithms. The environment was extended with a fault injection and VM placement algorithms to facilitate availability data generation with metrics. Findings from the availability comparisons of the four algorithms showed that algorithms that perform well on one metric perform poorly on the other metrics. Worst Fit algorithm, for instance, has scored the highest availability with respect to First Fit, Best Fit and Round Robin with full memory and CPU utilization. Although there is no clear winner, Worst-Fit and Round-Robin have shown better results than the First-Fit and Best-Fit algorithms.

Based on these findings, a conclusion is reached for VM placement algorithms highlighting the importance of thorough empirical studies for availability as one important aspect of cloud computing.

6.2 Future Work

This thesis work focused on heuristics-based bin packing VM placement algorithms specifically the greedy and round robin placements. For the future work, additional algorithms that are not compared from bin packing, as well as meta-heuristics, will be considered for better comparison. Furthermore, VM placement algorithms compared assume homogeneity of PMs but in realistic settings, this may not always be the case. Hence the algorithm comparisons will also be made to consider the heterogeneity of PMs.

References

[1] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, “Reliability and high availability in cloud computing environments : a reference roadmap,” *Human-centric Comput. Inf. Sci.*, 2018.

[2] H. Yang and M. Tate, “A Descriptive Literature Review and Classification of Cloud Computing Research,” *Commun. Assoc. Inf. Syst.*, vol. 31, no. 2, pp. 35–60, 2012.

[3] M. Nabi, M. Toeroe, F. Khendek, M. Nabi, M. Toeroe, and F. Khendek, “Author ’ s Accepted Manuscript Availability in the Cloud : State of the Art Reference : Availability in the Cloud : State of the Art,” 2015.

[4] M. Armbrust, A. D. Joseph, R. H. Katz, and D. A. Patterson, “Above the Clouds : A Berkeley View of Cloud Computing,” 2009.

[5] Z. Usmani, S. Singh, and S. M. Ieee, “A Survey of Virtual Machine Placement Techniques in a Cloud Data Center,” *Procedia - Procedia Comput. Sci.*, vol. 78, no. December 2015, pp. 491–498, 2016.

[6] H. Wu and C. Winer, “Network Security for Virtual Machine in Cloud Computing,” 2009.

[7] M. Masdari, S. S. Nabavi, and V. Ahmadi, “An Overview of Virtual Machine Placement Schemes,” *J. Netw. Comput. Appl.*, 2016.

[8] L. Fabio, “A Virtual Machine Placement Taxonomy,” 2015.

[9] Z. Á. Mann, “A taxonomy for the virtual machine allocation problem,” vol. 9, pp. 269–276, 2015.

[10] M. Filho, R. Oliveira, C. Monteiro, P. R. M. Inácio, and M. M. Freire, “CloudSim Plus : A Cloud Computing Simulation Framework Pursuing Software Engineering Principles for Improved Modularity , Extensibility and Correctness,” pp. 400–406, 2017.

[11] IEC 60300-1, “Dependability management – Part 1: Guidance for management and application,” pp. 1–7, 2014.

[12] J. Langston, “System Availability Benchmarking - A Survey.” unpublished.

[13] R. Jhawar, V. Piuri, and I. Universit, “Fault Tolerance Management in IaaS Clouds,” 2012.

[14] M. Nita, F. Pop, M. Mocanu, and V. Cristea, “FIM-SIM : Fault Injection Module for CloudSim Based on Statistical Distributions,” 2014

[15] I. Foster, “Virtual Infrastructure Management in Private and Hybrid Clouds,” 2019

[16] B. Gohil, “A Comparative Analysis of Virtual Machine Placement Techniques in the Cloud Environment,” vol. 156, no. 14, pp. 12–18, 2016.

[17] Y. Yu and Y. Gao, “Constraint Programming-Based Virtual Machines Placement Algorithm in Datacenter,” pp. 295–304, 2012.

[18] K. Mills, J. Filliben, and C. Dabrowski, “Comparing VM-Placement Algorithms for On-Demand Clouds,” 2011.

[19] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh, “Toward Virtual Machine Packing Optimization Virtual Cluster Management System : Grivon,” pp. 651–652, 2009.

[20] S. K. Mandal, “On-Demand VM Placement on Cloud Infrastructure On-Demand VM Placement on Cloud Infrastructure Master of Technology,” June, 2013

[21] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, “Study and Performance Analysis of Various VM Placement Strategies,” pp. 5–10, 2015.

[22] S. Dhingra, M. Madan, K. R. Babu, and V. Campus, “Comparative Study of Virtual Machine Placement Algorithms in Cloud Computing Environment,” vol. 3, no. 5, pp. 156–160, 2016.

[23] C. Practice and Z. Mann, “Which is the best algorithm for virtual machine placement optimization ?,” March, 2017.

[24] L. Shi, J. Furlong, and R. Wang, “Empirical Evaluation of Vector Bin Packing Algorithms for Energy Efficient Data Centers,” pp. 0–6, 2013.

[25] B. Liu, X. Chang, Z. Han, K. Trivedi, X. Chang, Z. Han, and K. Trivedi, “Model-based

Sensitivity Analysis of IaaS Cloud Availability," 2018.

[26] A. Brown, D. A. Patterson, and S. Hall, "Towards Availability Benchmarks : A Case Study of Software RAID Systems Computer Science Division University of California at Berkeley A General Methodology for Availability Benchmarking," 2000.

[27] D. Villegas, A. Antoniou, S. M. Sadjadi, and A. Iosup, "An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds," vol. 2, no. Section III, 2012.

[28] X. X. Liu, J. Qiu, and J. M. Zhang, "High Availability Benchmarking for Cloud Management Infrastructure," 2014.

[29] Tsidulko J (2017) The 10 biggest cloud outages of 2017 (So far). 2017; https://www.crn.com/slide-shows/cloud/300089786/the-10-biggest-cloud-outages-of-2017-so-far.htm, Accessed 1 Aug 2017

[30] Reliability HotWire, Relationship between availability and reliability https://www.weibull.com/hotwire/issue26/relbasics26.htm, Accessed October 2019

[31] Laerd Statistics, https://statistics.laerd.com/spss-tutorials/kruskal-wallis-h-test-using-spss-statistics.php, Accessed February 2020

[32] ITIL v3 (Information Technology Infrastructure Library)/Service Design, https://en.wikibooks.org/wiki/ITIL_v3_(Information_Technology_Infrastructure_Library)/Service_Design. Accessed October 2019

Appendix

Appendix A: Simulation result for availability comparison of algorithms where MTTR and MTBF are in minutes.

Worst Fit Algorithm

Availability(%)	MTTR(min.)	MTBF(min.)
62.14	92793	391368
100.00	14399	478826
57.48	5	494027
100.00	0	501419
97.84	105278	382858
78.41	6	487289
86.25	7183	477684
60.33	26175	453825
90.93	54779	439336
100.00	189091	292604
66.93	203124	277310
100.00	0	486089
58.77	0	486159
77.96	1	480238
85.83	98945	383551
90.77	57701	428296
32.64	2	483281
75.60	2	481267
94.00	0	486634
75.18	0	488481

First Fit Algorithm

Availability(%)	MTTR(min.)	MTBF(min.)
62.14	185124	303790
100.00	5	483508
57.48	208690	282082
100.00	5	484410
97.84	10838	489967
78.41	104999	381233
86.25	67553	423733
60.33	192854	293321
90.93	43927	440338
100.00	5	480164
66.93	161357	326504
100.00	6	483350
58.77	200082	285184
77.96	106326	375991
85.83	68450	414780
90.77	44438	437120
32.64	325546	157751
75.60	117349	363505
94.00	29035	454482
75.18	136195	412450

Best Fit Algorithm

Availability(%)	MTTR(min.)	MTBF(min.)
87.93	59719	435129
54.47	218799	261717
94.34	27210	453731
50.08	242511	243259
100.00	5	487565
88.07	58317	430662
79.61	98598	385024
84.29	75448	404690
81.44	91266	400521
100.00	2	481190
79.49	99036	383913
63.72	174175	305906
100.00	1	482266
100.00	3	489358
51.16	234642	245742
59.46	196695	288461
72.00	135530	348430
79.03	100759	379683
100.00	0	561006
92.48	36299	446084

Round Robin Algorithm

Availability(%)	MTTR(min.)	MTBF(min.)
85.41	70667	413549
100.00	0	485957
100.00	0	483400
63.99	173309	307924
57.46	204783	276644
46.37	261764	226297
100.00	0	485148
62.21	185436	305221
71.70	137655	348699
92.85	34702	450907
49.08	245411	236531
100.00	0	483981
100.00	0	482391
66.01	163173	316828
94.14	29902	480113
100.00	0	482147
84.31	75500	405790
100.00	0	484102
84.12	78388	415276
100.00	0	486434

Appendix B: Simulation result for availability comparison based on CPU loads with full memory utilization.

Worst Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
9.97	2736240	303018.1
23.36	2490622	759276.6
13.07	1814135	272656.6
14.10	1812246	297533.2
22.42	1761531	509084.2
24.60	1614535	526787.9
33.14	1380398	684083.5
22.49	1275799	370086.8
39.84	814536	539415.8
39.84	814536	539415.8
36.45	745674	427620
41.06	712682	496526.2
39.67	692017	455064
28.57	670970	268368.1
33.95	621592	319551.5
41.88	548152	395005.5
43.22	530731	404033.3
47.21	497231	444698.1
74.55	273870	802125.4
72.82	257748	690654

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
63.12	177234	303361.1
100.00	0	500470.1
98.58	6901	480268.9
100.00	0	484854.1
100.00	0	492894.8
100.00	1	595064.7
57.00	207310	274854.6
100.00	6	558050
100.00	0	487776.7
70.74	141842	342985.4
47.80	252334	231068.9
93.56	31193	453487.7
100.00	3	483830.3
100.00	0	489321.8
90.95	43765	439828.7
100.00	0	500470.1
59.13	196513	284275.4
60.35	190749	290274.6
100.00	0	488102
46.63	258568	225927.6

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
94.60	26344	461180.9
72.27	134592	350844.3
92.83	34802	450307.7
64.98	169982	315457.8
93.84	29766	453078.7
75.01	120342	361305.4
78.13	105125	375657.1
55.12	217000	266535.3
100.00	3	483786.2
83.74	78216	402887.1
100.00	0	488672.1
100.00	0	489289.9
47.70	255593	233124.4
47.18	258210	230638.1
100.00	0	513045.9
79.43	100571	388429.5
92.98	33810	447651.9
100.00	6	602618.9
100.00	4	489562.4
98.77	7624	610473.4

First Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
39.33	1282786	831504
38.86	800921	508992
38.69	828793	523033
35.00	1383242	744864
34.70	775435	412096
32.78	1033421	503910
32.30	1205107	574927
30.70	656948	291034
24.88	1481975	490835
21.31	1814941	491583
19.60	755826	184273
18.41	2362175	532865
17.72	2463180	530656
16.96	2663744	544186
14.32	1733623	289767
13.40	1727087	267152
11.79	2696540	360524
11.67	2703115	357113
9.94	2678905	295718
9.82	2685071	292328

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
60.12	191646	288898
92.60	46371	580444
81.09	92496	396623
54.25	303356	359654
57.27	390103	522750
100.00	0	697050
56.29	241686	311293
58.00	204951	283044
83.45	111759	563532
81.57	89287	395161
35.23	318840	173397
100.00	6	488127
53.49	227323	261410
100.00	3	487355
44.48	270276	216548
100.00	4	484986
43.34	513398	392640
92.18	37955	447623
100.00	4	487588
82.83	126139	608613

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
66.19	163952	321036
87.79	58787	422850
85.63	69001	411095
35.85	309595	173052
33.91	320215	164302
100.00	3	485137
41.21	285406	200055
65.86	165286	318811
80.41	96042	394302
86.45	65957	420814
36.12	313263	177116
100.00	7	480231
51.44	237523	251646
100.00	4	480312
36.98	305717	179412
100.00	5	492768
27.85	348363	134442
100.00	7	486978
100.00	6	480427
79.82	103871	410805

Best Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
51.06	458535	478405
43.49	1278369	983701
42.78	534249	399458
40.43	560468	380450
40.38	666837	451619
34.64	1283152	680138
32.68	634476	308057
26.85	688141	252549
25.36	697966	237089
25.06	1740216	581834
24.92	1476253	490106
24.57	1549418	504688
24.41	1219376	393739
23.19	1192470	360056
23.16	1733385	522510
22.37	1720818	495867
22.11	727538	206525
20.37	1759270	450112
19.21	1614613	383922
17.77	1725855	372877

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
59.8	343783	511414.3
68.39	165267	357497.7
100	0	571833.9
91.17	42586	439489.8
44.33	271958	216529.7
100	0	491482.5
52.89	409107	459278.3
76.18	115124	368158
100	0	485839.5
51.54	249597	265451.9
56.86	235055	309814.8
60.17	347567	525138.6
76.43	156767	508483.5
75.68	131525	409343
50.06	244049	244597.6
100	0	521095.2
58.47	201584	283867
44.41	268527	214520.2
75.4	119646	366654.4
100	0	482734.2

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
70.21	158823	374364
61.93	186136	302775
51.31	239585	252482
38.81	297969	189008
95.45	22165	465357
83.27	80444	400260
100	5	489076
96.21	18283	463980
100	5	487481
65.4	167126	315882
42.29	282855	207312
51.47	239466	253927
35.96	307699	172817
21.4	383549	104399
36.49	306876	176292
100	5	481968
44.77	268944	218021
54.74	218157	263851
59.91	193430	289074
79.08	102133	386133

Round Robin algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
26.98	871923	322202
8.85	2762444	268133
15.27	1793540	323302
43.19	784181	596132
21.98	1670687	470678
100.00	0	941368
37.21	764960	453236
9.04	2719237	270181
27.26	678968	254470
28.39	1465621	581102
18.44	1745051	394426
13.86	2589359	416685
100.00	0	941842
14.62	797420	136518
100.00	0	944260
12.29	2743593	384581
31.41	1520782	696459
33.56	1121773	566688
52.22	680042	743088
49.09	479056	461960

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	4	486680
52.18	230396	251363
74.42	125046	363757
100.00	0	495243
80.96	92184	392083
95.69	21099	468811
100.00	0	491599
68.54	151804	330652
47.48	253155	228849
100.00	6	484852
80.67	94773	395474
73.98	126854	360730
75.04	121747	366000
100.00	4	484441
71.04	140912	345731
67.77	155572	327166
82.67	83408	397849
100.00	5	483877
100.00	0	490674
84.18	76200	405361

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
89.29	51449	429065
26.02	356603	125414
69.15	149189	334383
91.19	43210	447301
77.21	111831	378775
100.00	1	484090
100.00	7	482257
61.02	189999	297477
100.00	5	480270
84.95	73007	412215
75.31	118896	362614
65.58	168365	320765
100.00	2	482703
90.11	48754	444076
100.00	2	483207
52.26	229985	251770
83.60	80080	408335
87.69	61900	440951
100.00	7	480552
100.00	2	484746

Appendix C: Simulation result for availability comparison based on Memory loads with full CPU utilization.

Worst Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
97.98	9871	477791
95.98	19449	463917
85.94	67926	415331
73.85	125909	355526
100.00	5	487234
79.99	120257	480668
56.64	210361	274767
100.00	6	484151
79.94	97842	389899
95.27	23041	464393
100.00	6	483499
100.00	7	486151
60.33	190868	290271
45.64	264313	221880
92.39	36656	445231
100.00	6	486056
100.00	6	482784
92.52	36790	455341
48.54	247875	233777
90.87	44435	442493

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
43.15	279866	212410
98.66	6685	492295
76.44	116504	377904
73.59	129032	359570
100.00	0	492819
80.38	94586	387408
71.88	135350	345897
100.00	5	489931
94.00	29371	459933
98.60	6923	487575
100.00	5	480735
100.00	7	494692
79.94	98751	393424
41.32	287733	202570
94.37	27154	455562
100.00	7	482509
100.00	3	483312
86.71	63784	416298
69.16	148801	333724
63.16	179459	307673

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	0	486374
92.82	34794	449814
100.00	0	485746
69.22	148526	334067
100.00	0	486571
69.79	145445	336062
66.42	164253	324923
100.00	1	480886
100.00	0	484730
91.26	42004	438426
100.00	0	489046
96.45	18241	495283
100.00	0	485328
46.53	261795	227854
72.23	133362	346937
96.82	15298	465731
100.00	0	487715
80.80	142885	601383
63.21	177576	305111
100.00	0	484696

First Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
76.65	112997	370909
59.50	195197	286716
100.00	6	488803
100.00	5	488717
19.46	391657	94612
100.00	3	497241
69.78	148388	342695
56.96	208582	276030
79.46	101021	390772
100.00	5	481206
66.54	162144	322375
34.74	316447	168474
56.96	207120	274096
57.21	206726	276421
61.36	185535	294670
86.66	64287	417754
89.23	51877	429969
93.51	31455	452843
69.78	148388	342695
68.73	150577	330892

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
72.48	132590	349126
97.94	10126	482166
59.33	195908	285770
58.44	199856	280973
100.00	5	487176
47.44	254501	229680
72.71	149780	399134
100.00	3	488088
70.35	199078	472263
52.59	232668	258129
82.22	86598	400507
100.00	4	487574
80.59	93603	388542
100.00	2	480897
84.01	77801	408898
69.99	145030	338195
67.11	161735	329963
61.52	186791	298663
73.07	131124	355858
100.00	3	488019

90 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
77.56	108998	376820
54.61	218625	262999
100.00	2	487996
100.00	1	486047
18.61	391817	89598
100.00	0	485999
65.49	169387	321498
36.80	308733	179731
80.54	94018	389150
100.00	0	486712
56.82	208923	274919
32.19	329232	156318
48.19	248746	231362
53.74	226879	263569
56.10	213031	272251
82.91	82139	398482
94.34	27503	458021
100.00	3	488013
62.63	180546	302543
57.29	207664	278545

Best Fit algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
96.25	18398	472450
63.60	204327	357052
63.49	176033	306086
73.20	130899	357483
74.08	126692	362117
96.16	18675	467703
96.22	18494	470922
98.49	7298	475543
84.12	76533	405559
41.80	280867	201708
66.42	161773	319949
89.68	51234	445230
86.98	63858	426534
58.99	197090	283538
77.40	110199	377474
57.89	203678	279983
90.33	46961	438917
100.00	5	486134
70.98	140185	342826
52.60	231909	257345

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	2	509586
37.76	302811	183711
60.23	194469	294488
76.81	112881	373797
78.34	104703	378618
97.53	12005	473243
60.15	194711	293878
100.00	1	486618
81.67	88195	392915
20.47	386939	99622
67.07	161152	328246
93.43	31631	449841
88.37	56244	427437
100.00	7	484793
79.22	100151	381704
40.66	290261	198884
96.23	18115	461932
100.00	1	484448
67.08	158650	323328
24.06	367698	116528

90% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	1	495353
43.06	277561	209884
55.76	218437	275341
78.27	106804	384636
84.15	77248	410004
100.00	7	486118
48.24	249526	232561
100.00	0	499579
89.21	52052	430253
28.74	347559	140146
71.70	138161	350086
96.44	18260	495229
94.84	25159	462133
100.00	3	482938
85.71	68737	412114
43.30	277556	211937
97.64	11374	471590
100.00	0	491455
75.09	122249	368430
35.17	312119	169294

Round Robin algorithm

10 % Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	2	490113
43.97	269611	211615
100.00	0	483927
40.62	285758	195503
67.51	156047	324289
100.00	0	483629
100.00	2	481641
96.82	15601	474821
100.00	2	485788
75.87	115859	364319
55.52	215418	268916
100.00	6	480484
68.58	153881	335937
78.95	101537	380894
100.00	0	483251
92.03	38695	446815
100.00	0	486828
35.87	308179	172404
100.00	0	491755
82.96	82805	403127

50% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
100.00	6	573984
66.88	162193	327473
100.00	3	485461
64.21	172143	308824
68.72	152936	335996
100.00	2	484494
100.00	5	486686
90.66	45016	437006
100.00	6	491940
72.32	134611	351624
66.94	160124	324289
100.00	7	485981
72.38	135301	354519
78.31	104504	377222
100.00	0	489974
89.80	51265	451141
100.00	3	485461
56.88	207382	273566
100.00	4	480062
86.96	62858	419222

90% Utilization Level

Avail. (%)	MTTR (min.)	MTBF (min.)
85.01	73172	414891
58.00	204210	282018
100.00	0	496269
53.39	225274	258094
68.54	152916	333154
100.00	0	490936
95.04	24309	465434
81.03	94732	404522
86.51	66075	423850
69.35	148666	336397
66.96	160123	324448
80.50	93969	387952
75.25	149691	455003
74.33	125321	362840
100.00	0	488623
76.72	113267	373176
100.00	6	482987
43.74	274466	213399
100.00	7	483923
76.43	115404	374169