Master of Science in Electrical Engineering
January 2018

Algorithms for efficient VM placement in data centers

Cloud Based Design and Performance Analysis

Mahammad Suhail Atchukatla

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Contact Information:
Author(s):
Mahammad Suhail Atchukatla
E-mail: maat16@student.bth.se
mahammad.suhail.94@gmail.com

University advisor:
Dr. Dragos Ilie
School of Computing

ABSTRACT

Content: Recent trends show that cloud computing adoption is continuously increasing in every organization. So, demand for the cloud datacenters tremendously increases over a period, resulting in significantly increased resource utilization of the datacenters. In this thesis work, research was carried out on optimizing the energy consumption by using packing of the virtual machines in the datacenter. The CloudSim simulator was used for evaluating bin-packing algorithms and for practical implementation OpenStack cloud computing environment was chosen as the platform for this research.

Objectives: In this research, our objectives are as follows

- Perform simulation of algorithms in CloudSim simulator.
- Estimate and compare the energy consumption of different packing algorithms.
- Design an OpenStack testbed to implement the Bin packing algorithm.

Methods:
We use CloudSim simulator to estimate the energy consumption of the First fit, the First fit decreasing, Best fit and Enhanced best-fit algorithms. Design a heuristic model for implementation in the OpenStack environment for optimizing the energy consumption for the physical machines. Server consolidation and live migration are used for the algorithms design in the OpenStack implementation. Our research also extended to the Nova scheduler functionality in an OpenStack environment.

Results:
Most of the case the enhanced best-fit algorithm gives the better results. The results are obtained from the default OpenStack VM placement algorithm as well as from the heuristic algorithm developed in this simulation work. The comparison of results indicates that the total energy consumption of the data center is reduced without affecting potential service level agreements.

Conclusions:
The research tells that energy consumption of the physical machines can be optimized without compromising the offered service quality. A Python wrapper was developed to implement this model in the OpenStack environment and minimize the energy consumption of the Physical machine by shutdown the unused physical machines. The results indicate that CPU Utilization does not vary much when live migration of the virtual machine is performed.

Keywords: Bin packing, Energy Consumption, Live migration, OpenStack, , Virtual Machines.

ACKNOWLEDGEMENTS

It is a great pleasure to express my deepest thanks and gratitude to my supervisor Dr. Dragos Ilie for his exceptional guidance and support during the entire duration of my thesis work. I would also like to thank him for giving me an amazing opportunity to learn and develop skills voluntarily apart from regular courses and assignments. His comments and encouragement has helped me throughout my thesis for the successful completion.

I would like to thank Dr. Kurt Tutschku, our thesis examiner for conducting and managing this course. I would also thank him for his lectures and motivational discussions throughout the thesis course.

I would like to thank the Department of Communication Systems for this educational opportunity which has tested and pushed me beyond my abilities.

I would also like to thank my friends and colleagues at the University for their encouragement and moral support which made my stay and studies more enjoyable. Many thanks to my family for their unconditional love and support during last two years.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The decision on which physical machines (PM) to place each virtual machine (VM) is very important for the efficient cloud operation. On one hand, it is important that VMs obtain adequate resources (e.g., CPU, memory, network) from the hosting PM so that its performance is not impaired. On the other hand, the cloud operator would like to consolidate VMs on as few PMs as possible to maximize resource utilization and lower energy consumption. Efficient VM placement is like the vector bin packing problem, which is known to be NP-hard[1].

Another problem is that resource consumption in a VM is not constant but rather changes dynamically over time. One way to address this is to place VMs based on peak resource usage. However, this is a wasteful approach because peak usage occurs rarely[2]. Another option is to migrate VMs to PMs with more resources when necessary, for example through live migration. Unfortunately, live migration requires additional resources on its own and thus contributes to the system-wide load[3] [4]. This means that not only the migrating VM is affected, but also other devices on the migration path.

It is desirable that a placement algorithm strikes a balance between stability of placement and efficient resource usage (i.e. packing) while considering the cost live migration. Also, it was shown in [5] and [6] that the QoS performance is sub-optimal unless a QoS goal can be expressed explicitly in terms of a workload independent QoS metric[7].

1.1 Motivation

Cloud computing is a revolutionized technology in information and communication industry. It continuously increases demand for resources in different organizations services like augmented reality, self-driving cars and 5G mobile services. So, the demand for datacenters keeps growing in around the world. However, cloud datacenters consume a high amount of energy it also emits the heat and carbon dioxide in the environment. According to a report of “Total Consumer Power Consumption Forecast,” it predicts that in 2025 datacenters consume more than 20% of the total power in world and data centers also responsible for 3% of total greenhouse gas emission [8].

This is a very big problem for large size tech industries like Facebook, Google, Amazon, Alibaba, etc. To eliminate inefficiency and wastage of power consuming resources can be done by improving both infrastructure of the data center and effective management tools using resource allocation algorithms. They are investing a huge amount of money on minimizing the resource utilization with different methods.

1.2 Research Questions

1. **How can energy consumption be optimized by using bin packing placement algorithm?**

2. **How should a heuristic model be defined to provide an efficient method to optimize the energy consumption in the data center?**

3. **How is the overall Number of VM migrations of node affected when performing bin packing algorithm?**

1.3 Aims and Objectives of the Thesis

The main aim of this thesis is to investigate the efficient algorithm for virtual machine placement (Bin placement). We use CloudSim simulator to estimate the power consumption of the real datasets. Implement this model in an OpenStack cloud platform.

The main objectives of the thesis are as follows:

- Estimate the power consumption of data centers using CloudSim simulator using placement algorithm.
- To design an OpenStack testbed and implement Bin packing.
- To design a heuristic algorithm to evaluate the performance metrics such as degradation of resources and power consumption.
- Increase the packing efficiency of VMs with minimal QOS.

1.4 Outline of the Document

- Chapter 1: This chapter describes the basic overview and introduction of the problem which we are solving.
- Chapter 2: Background of the topics like cloud computing and virtualization technologies.

- Chapter 3: continue with an introduction to cloud computing, Virtualization concepts and the tools used.

- Chapter 4: Presents the detailed description of the related work

- Chapter 5: Present the method to discussed problems and present implementation of the framework.

- Chapter 6: Provide results and analysis of the solution.

- Chapter 7,8,9: Present the conclusions and discuss the future work.

2 CONCEPTUAL BACKGROUND

2.1 Cloud Computing:

Cloud computing is a large pool of systems are connected public or private network for providing the infrastructure for storage, applications, and data to the user. In cloud computing, a single PM can host several VMs simultaneously. The rapid growth in demand for computation power of resources shifts toward cloud computing model such as large-scale virtualized data centers around the world. The cloud computing model provides the customer to pay per usage basis. Instead of purchasing the infrastructure and the IT service upgrades, many organizations are using cloud-based operators such as Amazon Web Services, Google Cloud, and Rackspace platforms. These data centers consume a huge amount of electrical energy during operation.

Cloud providers offer services into three categories.

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a service (IaaS)

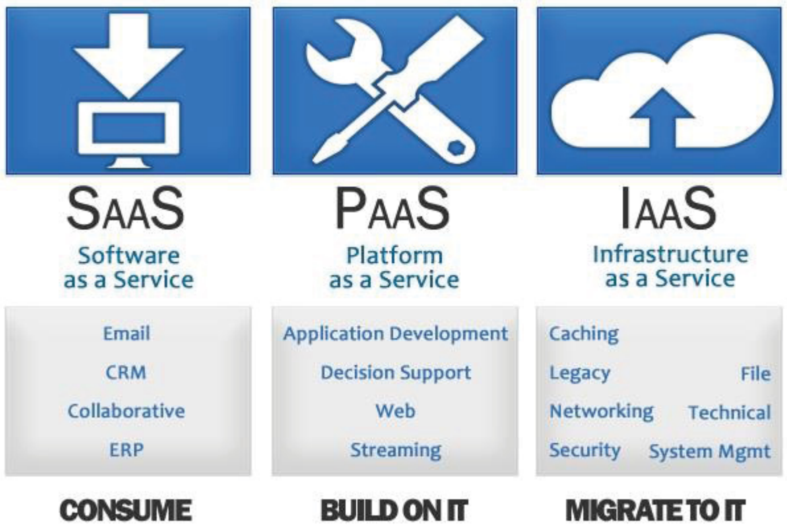

Figure 2.1 Block diagram of cloud computing [9]

Nowadays, data centers embrace the virtualization technology by implementing their services in many VMs hosted in a pool of PMs. It is important that careful allocation of the physical resources to VMs so that service performance is not impaired and energy consumption is limited. The algorithms used for allocating virtual machines placements impact on multiple factors.

1. Energy consumption of PMs
2. Performance of VMs
3. VM migration

2.1.1 Energy consumption of PMs

A good VM placement algorithm minimizes the number of PMs used for utilizing all VMs without degrading the performance of the deployed applications. Energy consumption is not constant all time. Dynamic voltage and frequency scaling can be used based on a load of PM. Dynamic voltage scaling increased if a load of PM increased(overvolting) or Dynamic voltage scaling decreased if a load of PM decreased(undervolting)[9].

2.1.2 Performance of the virtual machine

To an aggressive VM, placements may cause application performance degradation like consuming high electric energy, emits high noise and consume high memory [4]. In that case, the host PM cannot accommodate the number of resources needed by the virtualized application. Which in turn likely leads to violation of service level of the agreements (SLAs) depending on the terms of customer and service provider. Too much service level violation may cause penalties and also loses customer satisfaction.

2.1.3 Virtual machine migration

Live migration can lower VM downtime and leads to high resource utilization during migration. That means if the number of VM migration increases then energy consumption also increases. Therefore, effective placement algorithms should aim to minimize the number of VM migrations[3].

2.2 Cloud platform types

Cloud computing is available to consumers in three types.

1. Public cloud
2. Private cloud
3. Hybrid cloud

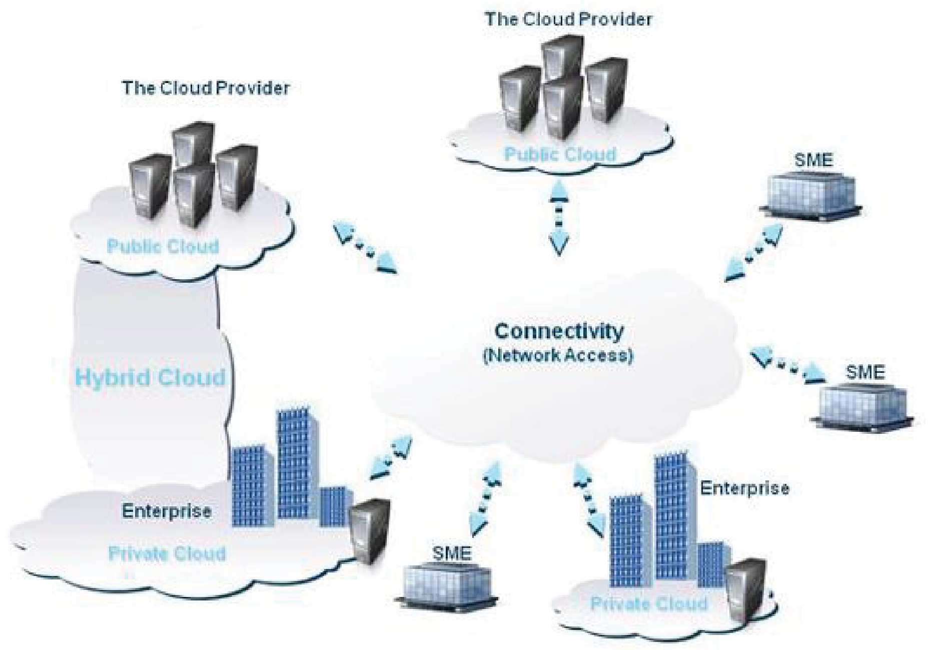

Figure 2.2 Overview diagram of cloud classification

Depending upon requirements and functionality these are classified into different levels of security, management, and availability. OpenStack environment can be used as public, private and hybrid cloud models.

2.2.1 Public cloud

A public cloud is basically using the Internet to provide features like computational resources, applications, and storage to consumers. Examples of public clouds include Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Windows Azure Services Platform, Sun Cloud, and Google App Engine.

This type of cloud is associated with high user satisfaction and cost efficiency. Because provider can bare all resources like hardware, with security [10]. Public cloud follows pay per usage rule so that user can pay only for the resources they have used.

Some limitations are also listed in the public cloud. It is not useful for some organization because they own in terms and condition for sharing internal data.

2.2.2 Private cloud

Private cloud denotes a data center under the ownership and control of the individual organization. This cloud will have full control over flexibility, security, scalability, automation, and monitoring to the organization. The main aim of the private cloud is not to sell "a service" to external customers instead to gain a benefit to maintain their own data centers.

Private cloud data centers are expensive because the organization needs to buy entire infrastructure for deployment and oversee the day-to-day operations. So, this cloud is not fit for small-scale industries due to a limited budget.

2.2.3 Hybrid cloud

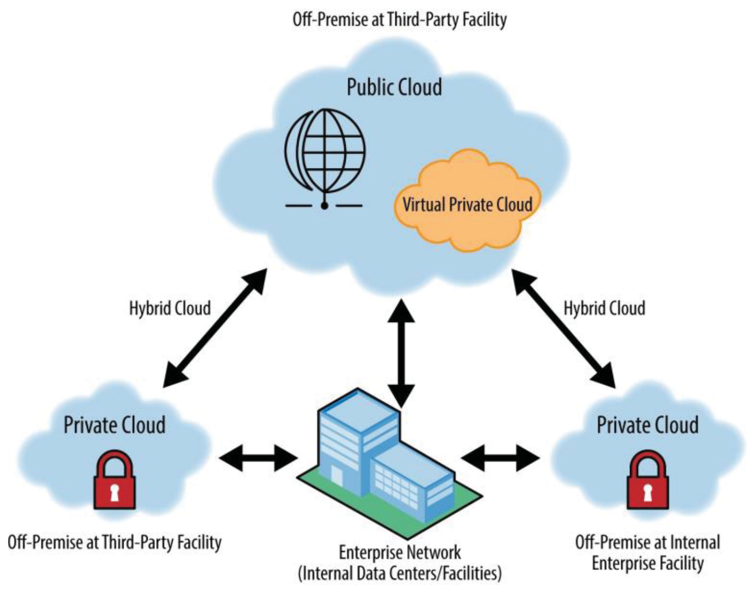

Figure 2.3 Overview diagram of a Hybrid cloud

Hybrid cloud is an integrated cloud service using both public and private cloud infrastructure to perform the operation within the organization.

Public cloud is given better scalability with limited boundaries because resources are used by the cloud providers. Using private cloud depends on configuration and resources it gives limited scalability. By using the hybrid cloud, it gives the benefit from public cloud scalability while reducing the private cloud demand. So, it became the greater cost efficiency compares with the public cloud.

2.3 Virtualization

Virtualization is technology that allows creating multiple virtual environments using single physical hardware. In this technology, Hypervisor has the ability to split physical machine resources to virtual machines appropriately. It allocates resources to virtual machines like memory, graphics, Networking parameters and virtual cores used for VMs. This technology is continuously developing to minimize the wastage of the physical machines resources.

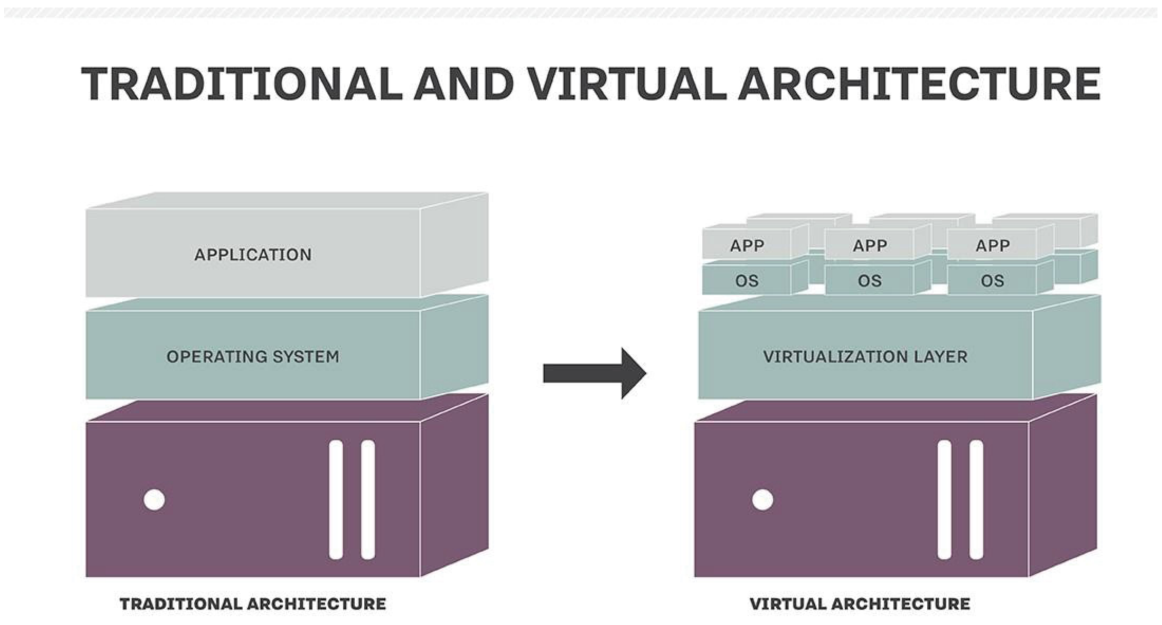

Figure 2.4 Virtual machine architecture

2.4 Type of virtualizations

2.4.1 Network visualization

Network virtualization is a key and future of the cloud computing. A computer network starts with network interface card which connected with layer 2 network to a layer 2 (L2) network (Ethernet, WiFi, etc.) segments.

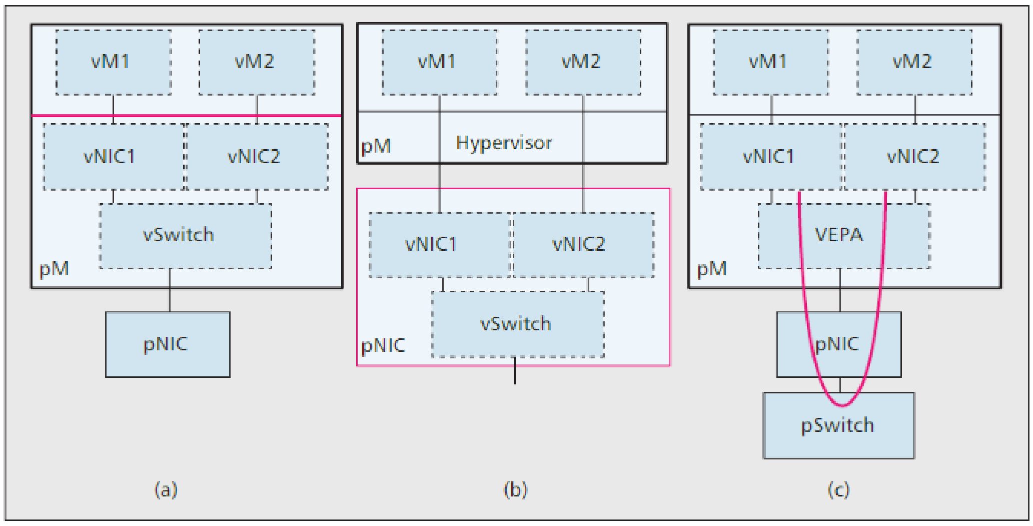

Figure 2.5 Block diagram of network virtualization

Several L2 network segments may be interconnected via switches (a.k.a. bridges) to form an L2 network [11]. If we are running the multiple VMs on physical machine it should have at least one virtual NIC as shown above figure. One way to solve this issue by adopting the virtual switch in the hypervisor. We use notation P means physical V means virtual.

2.4.2 Server virtualization

Server virtualization is mostly targeting to minimizing the resource utilization of the new generation data centers. In server virtualization, the virtual layer is placed the top of the physical layer[12].

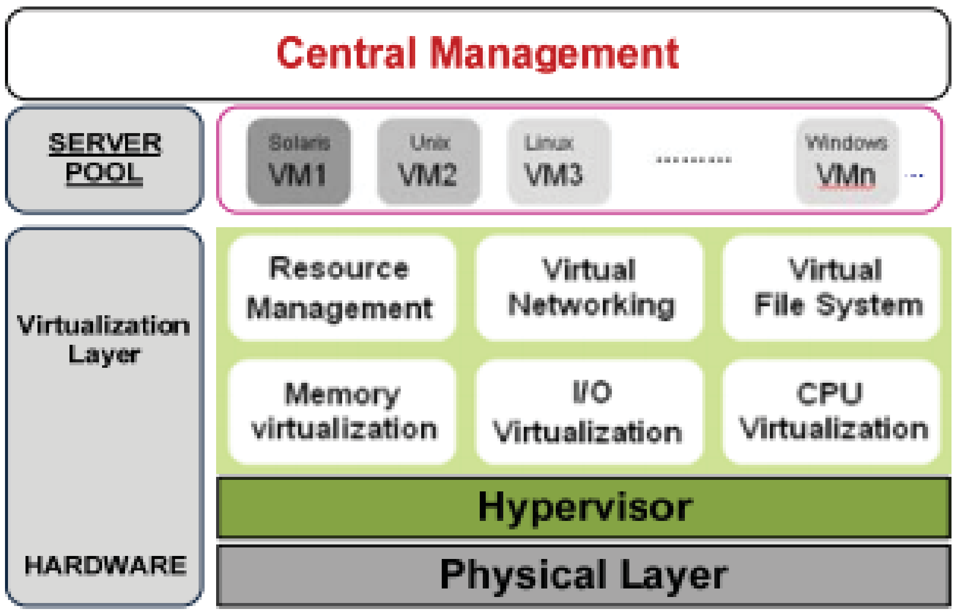

Figure 2.6 Block diagram of Server virtualization

The server virtualization achieves to enable the use of multiple virtual servers(VMs) on physical servers. Here each virtual server running its own operating system by using physical resources[13]. The most popular software is VMware, Microsoft, and Citrix.

2.4.3 Desktop virtualization

Desktop virtualization system generally contains the following four specific modules called server infrastructure module, virtual desktop host module, connection agent module and thin client module[14].

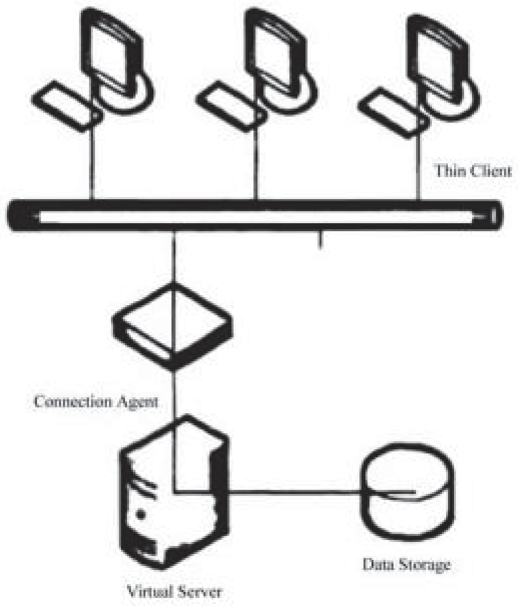

Figure 2.7 Overview diagram of a Desktop virtualization

The above figure shows the Server infrastructure module perform the data backup, Resource allocation, and application deployment. The virtual server used for creating a server for the desktop operating system. Connection agent provides the remote access to the user to connect operating system and the client module is the Virtual application of desktop available for the user.

2.5 Advantages using Virtualization

1. In virtualization, it maximizes the utilization of physical machine resources.
2. Downtime reduced in upgrades.
3. Security increases writing individual security policies by separated firewall rules in each virtual machine instead of a physical machine.
4. Import and migrate virtual machines very easily.

2.6 Bin Packing:

Virtual machine placement is important to resource and power management in cloud computing. It mainly deals with both the virtual machine and physical machines. A good virtual machine placement minimizes the overload of the resources and also energy consumption can be reduced. the effectiveness of the virtual machine placement measured by packing efficiency (Tightness of the packing). Resource allocation of the virtual machine is done in two stages

1. Each virtual machine assigned to host by their resource utilization of memory utilization, disk space, and communication bandwidth.
2. In the second stage, virtual machine allocation to the physical host by implementing the packing algorithm.

The One-dimension bin packing problem is defined as a set of variable size i elements are pack all items into N containers with each container capacity C [15]. The bin packing problem has a large influence on the industrial sector, for example in scheduling applications (planning advertising spots, scheduling tasks on processors, scheduling tasks with resource constraints) and in transport field (loading trucks with weight constraints or with volume constraint, filling containers, pallets)[15].

Bin Packing is objects of different volumes must be packed in a finite number of the Bins or containers. In this approach objects are the virtual machines and bins are the physical machines. For minimizing bin packing problem, we choose the algorithm which gives the least number of containers used. There are many variations for solving bin packing problem like 2D bin packing, linear packing, packing by weight, packing by cost. The main aim of this Bin packing is to minimize the number of physical machines being used.

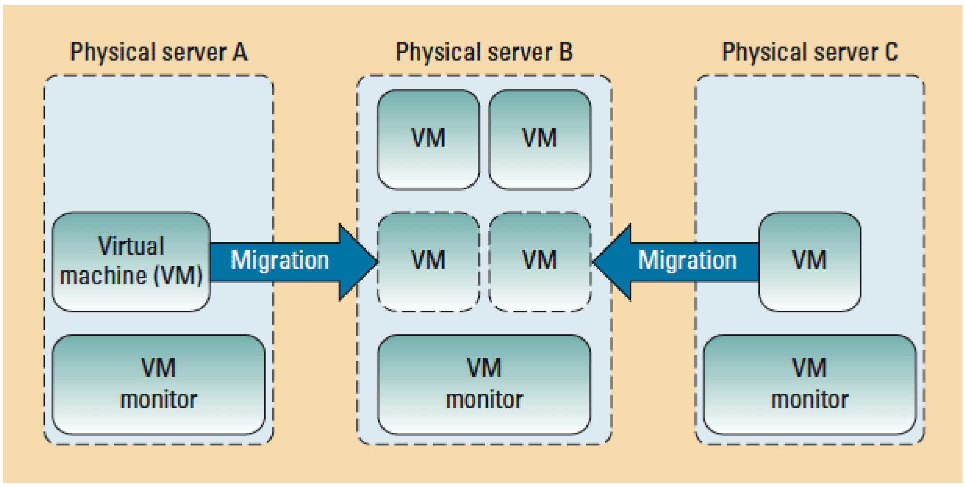

Figure 2.8 Block diagram of the server consolidation

The bin packing problem is mainly used in the live migration of the virtual machines. Most of the virtual machine migration strategies used are the Best fit, First fit and Next fit respectively.

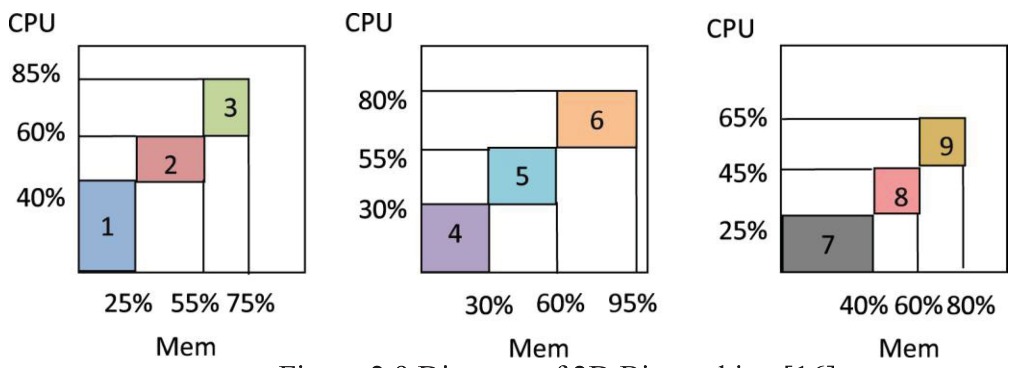

Figure 2.9 Diagram of 2D Bin packing [16]

In two dimension of the bin packing problem, each container can handle a rectangle which specified by width and height. In this bin packing we are given a set N= {N_1, N_2 ...., N_i} of I items, a capacity C ∈ I and size of functions s: N → I. The goal is to find the minimum the number of the bins used. A feasible assignment of into the items into I bins is a partition P_1, P_2, P_3, ..., P_N. such that for each P_k , the sum of the size of items P_k is not exceeded to not more than the capacity(C).

3 A SKETCH OF THE TECHNOLOGY

This section gives the brief idea about the technical applications used for this thesis work. The areas are covered in the following the order.

- CloudSim simulator
- OpenStack environment
- Load Generator

3.1 CloudSim Simulator

CloudSim simulator is a framework for modeling and simulation of cloud infrastructure and services [16]. This simulator primarily developed by researchers at the cloud's computing and distributed systems laboratory, University of Melbourne. CloudSim simulator mainly developed by using Java and it is publicly available in LGPL license[17].

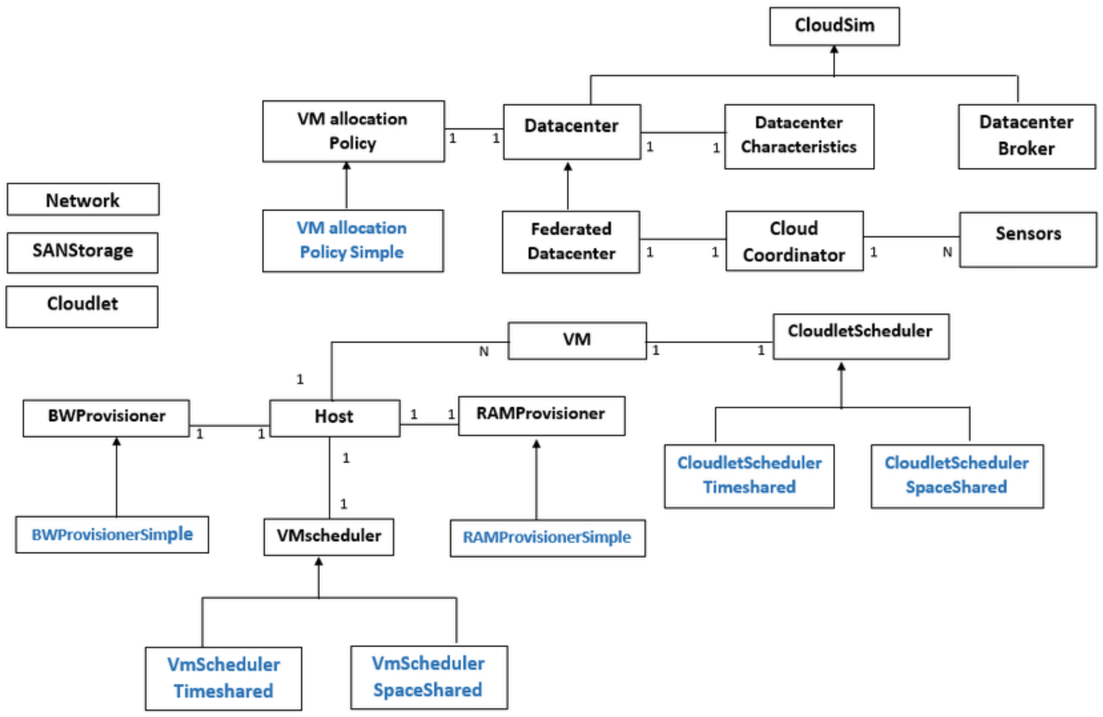

Figure 3.1 Overview diagram of a CloudSim simulator architecture

The main Advantages of cloud sim simulator:

- Support the simulation for the large-scale data centers.
- Support simulation for energy-aware computational resources.

- Support simulation for extending the VM allocation policy and Host allocation policies.
- Support simulation of network topologies and message passing application.
- Easy implementation by extending the user-defined cloud policy for the cloud.

The disadvantage with CloudSim simulator is it does not support the parallel process. It has limited functionality in host and VM network related parameters. In CloudSim simulator we can provide only bandwidth of each host and it is not possible to provide individual components (switch, links). However, it is not affected to this thesis experiment. Alternative simulation tools for CloudSim are CloudAnalyst, GreenCloud, NetworkCloudSim, EmuSim, SPECI, and GroudSim. Most of these tools are an extension of the CloudSim simulator.

3.2 OpenStack

OpenStack is a collection of the open source software projects that can be operated by cloud infrastructure. This is mostly deployed as infrastructure-as-a-service (Iaas). OpenStack is developed by the joint project by NASA and Rackspace in 2010. Presently it is managed by OpenStack foundation, a nonprofit corporation contain more than 500 companies. OpenStack has a new release approximately every six months. It is mainly written in Python language and relies on external libraries.

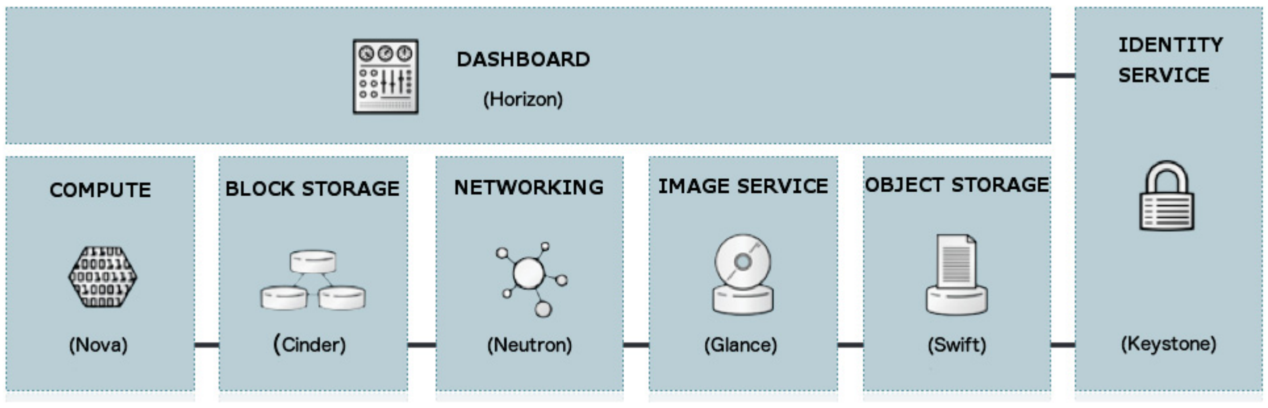

Figure 3.2 OpenStack service overview diagram [9]

OpenStack Components:

1. Nova
2. Neutron
3. Cinder
4. Keystone
5. Glance
6. Swift
7. Horizon
8. Heat

OpenStack has few other optional services which can improve the user experience. Optional components are as follow below

- Ceilometer (Telemetry)
- Heat (Orchestration)
- Trove (Database)
- Sahara (Elastic Map Reduce)
- Ironic (Bare-Metal Provisioning)
- Zaqar (Messaging Service)
- Manila (Shared Filesystems)
- Designate (DNS Service)
- Barbican (Key Management)
- Magnum (Containers)
- Murano (Application Catalog)
- Congress (Governance)

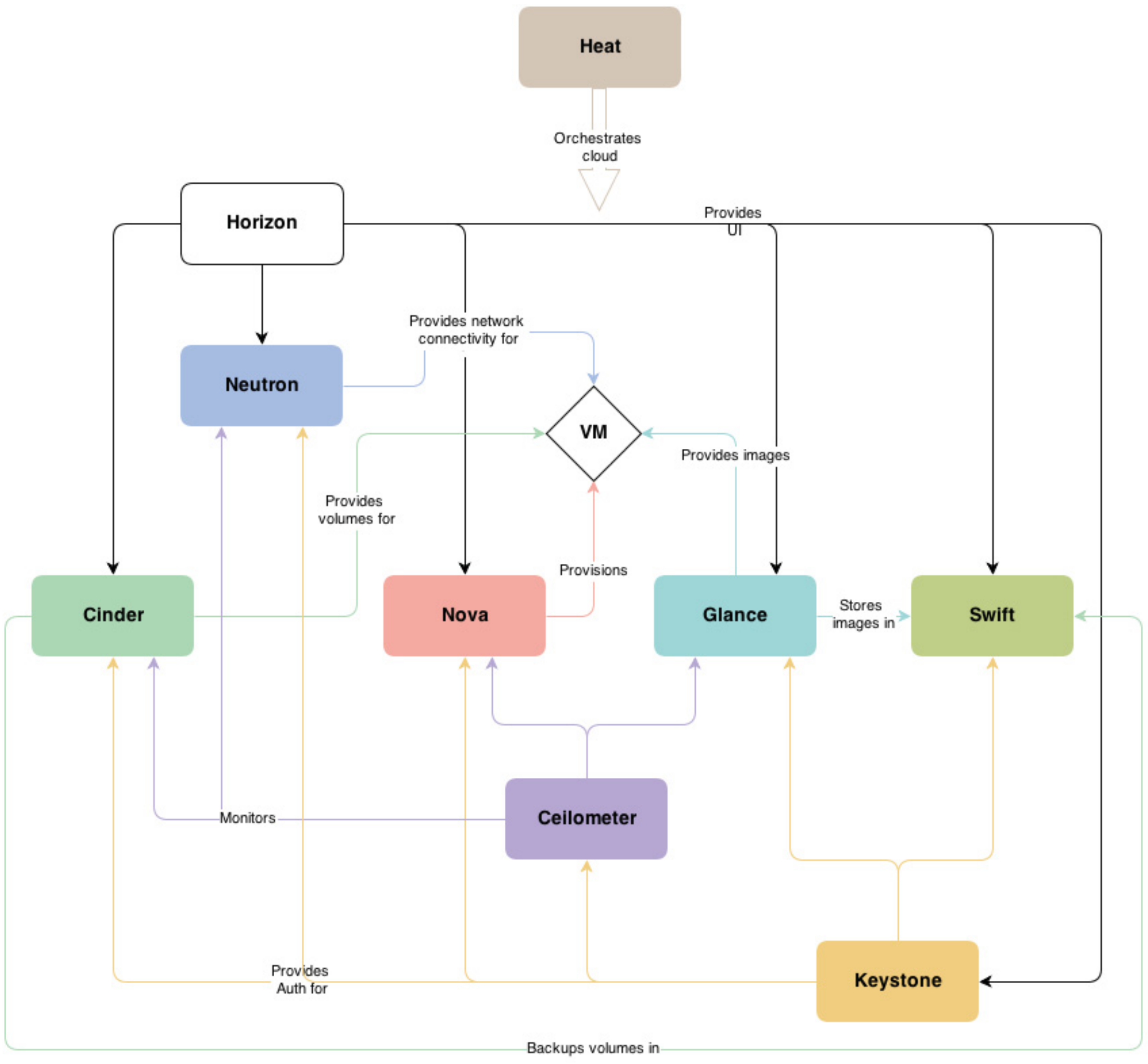

Figure 3.3 Diagram of OpenStack architecture

3.2.1 OpenStack service overview:

3.2.1.1 Nova

OpenStack Compute(Nova) is controller for the OpenStack environment. Nova mainly works with available virtualization technologies such as KVM, VMware, and Xen available hypervisors. This written in python and uses external libraries like SQL Alchemy(Database), Kombu and Eventlet (concurrent programming). In addition, RabbitMQ is used for the messaging services between the components.

3.2.1.2 Neutron

OpenStack Neutron is mainly used for networking tasks in the OpenStack environment. It is mainly used for allocating IP addresses, load balancing, firewalls and virtual private networks. Neutron managing IP address and manage the static IP address using DHCP. The dynamic Ip address has routed the traffic within infrastructure during maintenance and failure cases.

3.2.1.3 Cinder

Cinder is Block storage service in the OpenStack environment. It provides the persistent disk to virtual machines in the cloud environment. This gives virtualizes the management of block storage devices.

3.2.1.4 Keystone

OpenStack Keystone provides the user identity to OpenStack environment and enhances the security of request without outside of the project. This service mainly based on token-based authentication. Keystone managing the user account details, project information, group information and internal policy of the organization.

3.2.1.5 Glance

OpenStack Glance project mainly deals with image service discovery, server images, and image registrations. It is also the ability to store an instance server to snapshot image so that it can be used for creating a new server within a short time. Glance images stored in varies storage places like a File system, Swift using glance API and location will be stored in Glance registry.

3.2.1.6 Swift

OpenStack Swift is one of the major project developed by Rackspace and NASA during the initial OpenStack releases. The main functionality of swift includes the backup, storage and archiving unstructured data including the files, images, documents and virtual machine images.

3.2.1.7 Horizon

OpenStack dashboard provides the web-based user interface which integrates OpenStack components like Nova, Glance, Cinder, Heat, etc. It mainly designed by using Django and manage entire compute using a single application.

3.2.1.8 Heat

Heat is a template-based orchestration for utilizing OpenStack API calls to communicate cloud-based application in OpenStack. This template describes each parameter in service which is readable and understands to humans. This template primarily uses for the managing components in OpenStack but it is also extending to management tools like Ansible and Puppet.

3.3 CPU Load Generator

We need to execute the CPU load generator virtually in compute node for performing migration of virtual machine. Whenever CPU load passes the threshold value wrapper script will be executed. For CPU load generation there a lot of open source tools available on the internet. In this thesis, we used CPULoadGenerator which is available on Github [18]. The main advantage of this tool is it generates the fixed amount of load for a finite amount of time sample with selected cores.

```
Python CPULoadGenerator.py -l 0.5 -d 20 -c 1
```

In an above command, the script executes 50% load in 20 seconds duration in a single core.

4 RELATED WORK

2D Vector Bin packing problem is proposed by Chekuri and Khanna (1990) and proved the bin packing problem is APX- hard. Sunil [19] et el proposed the concept of the combining the memory and CPU utilization for solving the bin packing problem. In this paper provide a detail description of comparing SLA violation, Energy consumption during migration of VM.

Vector bins of variable sizes is an important problem that captures the human attention. A generalization for this problem was introduced et el Michael [20] which correlates the generation of feasible solutions for virtual machine placement problems. The paper also talks about the various additional constraints considered and the decomposition of a big problem into smaller fragments.

Server consolidation plays a very important role in efficiently managing the Cloud datacenter and optimize cost and energy consumption. The conventional process of server consolidation was performed by replacing the physical servers with the virtual machines. But, et el Sunil [19] introduces a new technique for server consolidation namely RFAware Server Consolidation that performs residual resource defragmentation which reduces the residual resources and in turn opens the door for new virtual machine allocations.

The enhancement of the performance of cloud resource allocation and virtual machines consolidation was triggered Et el Huda [21] by using a hybrid model by combining the existing genetic algorithms namely Combinational Ordering First-Fit Genetic Algorithm (COFFGA) and Combinational Ordering Next Fit Genetic Algorithm (CONFGA). The results show that the proposed Hybrid model outperforms the previous genetic algorithms. Et el Mohamed [22] also proposes a hybrid genetic algorithm using Best Fit Decreasing (BFD). The results prove that the deployed model gives the desired results.

The static allocations of the resources in virtual machines were not efficient and always reduces the efficiency of the entire system. Using dynamic allocations of the resources in virtual machines might solve the problem of such efficiency. Et el Zhuo [23] proposes a scheduling algorithm namely, virtual machine dynamic forecast scheduling (VM-DFS) that analyses the history of the memory consumption and decides the allocation of the resources based on the estimating the future requirements which reduced the number of physical machines. Several estimations were performed on the virtual machine requests and the normalized results show that about 17.08 % of physical machines can be saved.

Several Virtual machines were deployed on a single physical machine to increase the efficient utilization of the physical device and to reduce the cost and compromise with respect to performance to avoid buying new physical devices which are costly. But as mentioned earlier the dynamic allocation of the resources for the virtual machine will significantly reduce the number of additional physical blocks. Et el Chowdhury [24] gives the optimized algorithms that reduce the number of resources accordingly and occasionally deactivate the hosts if the resources are underutilized, thereby reducing the power consumption. A technique for clustering the VMs was also proposed which enables a new path for migrating the resources by taking feedback from CPU and RAM.

A prominent approach to consolidating the server with virtualization technology for optimizing the data center efficiency. Playing with the allocation and reallocation of resources for each virtual machine will result in reaching the performance targets of a physical machine but these lead to some limitations on system management, especially while dealing with large-scale data centers. An optimized technique for managing the resource wastage, power consumption and thermal dissipation costs simultaneously was developed et el Jing [25] which shows commendable results comparing to other techniques and approaches.

A performance analysis of Reordering Grouping Genetic Algorithm (RGGA) in different scenarios was explained by et el David [26]. It shows the efficiency and effectiveness of this algorithm which considering and testing in single-constraint scenario followed by multi-constraint scenario and a comparison of both the approaches were done.

Network bandwidth plays a major role for controlling the data traffic over several virtual machines in a physical machine that leads to performance degradation. Et el Jianhai [27] proposes a new method that opens door to solve the network bandwidth issue namely, Affinity-Aware Grouping method for Allocation of VMs (AAGA). An experimental setup of using virtual machines on a physical machine with heuristic bin packing algorithm was deployed and tested. Results show that the AAGA outperforms NAGA.

5 OVERVIEW OF THE APPLIED METHODS

Bin packing is NP-hard problem. So, this thesis works classified into two parts. These are a Theoretical approach (CloudSim) and practical approach (OpenStack environment).

Steps for the implementation:

1. Perform simulation of the algorithms in CloudSim simulator with different datasets.
2. Set threshold values and run Python wrapper script.
3. Store node metrics in local(VM) and global (Host) parameters in the database.
4. Choose the better packing algorithm for VM placement in the OpenStack environment.

5.1 Theoretical approach (CloudSim implementation)

In order to get the relevant results, it is very important to test model in real data. By default, CloudSim simulator uses beloglazov algorithm for placement of VMs. PowerVmAllocationPolicyMigrationAbstract.optimizeAllocation class extend for implementing Bin placement algorithms in CloudSim simulator. It is not so easy to compare all algorithms in side by side by variable data. So, we are using fixed parameters for testing this algorithm with different datasets. The main objective of this thesis is minimizing the resource utilization of the PM without affecting the SLA. So, we fix migration policy as "minimum migration time" during placement.

CloudSim simulator by default, it contains the planet lab workload trace. For both Google cluster, Bitbrain parameters and index of the parameters are different. So, we need to convert Google cluster and Bitbrain dataset to the required YAML input format. After conversion of the dataset, we provide the output file to the simulator. For comparison of the algorithms, we need to have the fixed parameters in both host and VM configuration.

5.1.1 Workload datasets

5.1.1.1 Google cluster (12000 PM)

Google cluster workload dataset is collected from the Google compute cell (kubernetes instance cluster). This Google cluster is a set of machines packed into racks and connected with high bandwidth cluster network. This dataset contains the 29 days' workload traces starting from May 2011 containing 12,500 machines. Below table explains the parameters which contain the dataset in both task event and job event table.

Parameters in the dataset:

1. ID
2. Start Time
3. Cores
4. CPU usage (MHZ)
5. Memory usage(KB)
6. Memory provisioned (KB)
7. Used CPU

5.1.1.2 Bitbrains (1750 VMs)

This dataset is distributed datacenter from Bitbrains(Dutch service provider) which managed hosting and business computation for enterprises. Bitbrain workload traces are two parts which are fast storage trace and Rnd workload trace. Fast storage workload trace collected 1250 VMs data period of the 1 month with total 4057 cores. Where Rnd workload trace collected 500 VMs data period of 3 months with 1444 cores. Each VM parameters stored individually in .csv format[28].

Parameters in the dataset:

1. Timestamp
2. CPU cores
3. CPU capacity provisioned
4. CPU usage (MHZ)
5. Memory provisioned (KB)
6. Memory usage(KB)
7. Disk read throughput(KB/s)
8. Disk write throughput(KB/s)
9. The network received throughput(KB/s)
10. Network transmitted throughput(KB/s)

5.1.1.3 Planet Lab (1000 VMs)

A set of utilization traces Planet lab VMs collected during 10 random days in March and April 2011. By default, planet lab workload trace included in Cloud Sim simulator. The data contains the CPU utilization of the thousands of the virtual machines located at more than 500 places around the world. In this dataset CPU utilization of each virtual machine is measured in 5 minutes.

Parameters in the dataset:

1. ID
2. CPU cores
3. CPU capacity
4. VM RAM (KB)
5. Bandwidth (KB/s)
6. Disk capacity (KB)

5.1.1.4 Constant Dataset (300 VMs)

This dataset is generated by the python based program using the pyyaml library. In this dataset, CPU load is generated by a constant value like 50%, 30% CPU utilization. We can generate the dataset by parameters like RAM, cores, CPU capacity, etc.

Parameters in the dataset:

1. ID
2. CPU cores
3. CPU capacity
4. VM RAM (KB)
5. Bandwidth (KB/s)
6. Disk capacity (KB)

5.1.1.5 Raw dataset

It also has same parameters as a constant database. This dataset generates dynamically change the policy of utilization by periodic, random, constant, decreasing pattern.

Parameters in the dataset:

1. ID
2. CPU cores
3. CPU capacity
4. VM RAM (KB)
5. Bandwidth (KB/s)
6. Disk capacity (KB)

5.2 Data Set Preparation

There is two type of the inputs we need to provide to CloudSim simulator. This is mainly followed by YAML(a human-readable data serialization language) file format.

I. Host resources
II. VM resources

Input sample for host resources:

```
- Host:
   BandWidth: 1000000
   RAM: 4096
   Cores: 2
   ID: [3]
   PowerModel: *id002
   DiskCapacity: 1000000
   CpuCapacity: [2660]
```

Input sample for VM resources:

```
- VM:
   StartTime: 3685042000
   Cores: 1
   DiskCapacity: 0
   Num: 1
   ID: [0]
   MIPS: [11703]
   RAM: 67108
```

5.3 A practical approach (OpenStack implementation)

5.3.1 Experimental Setup for OpenStack environment

The Experiment for evaluating the Bin packing of the virtual machines setup involved the use of 5 nodes provided through Mirantis platform.

The specifications of each server are:

- 4 compute nodes
- 1 controller node
- 8 Core processor
- 16 GB RAM
- Ubuntu 16.04 (Xenial) OS
- 256 GB Storage

OpenStack is a very powerful cloud-based environment. It is mainly classified as computing, controller, and network Node. In this practical approach compute node is like the physical machine and inside the virtual instance is a virtual machine. Finally, VM consolidation will be done according to the placement algorithm.

5.3.2 Data storage

Two types of data storage are used in this implementation. Data storage-1 deployed in every compute node so that it can collect CPU utilization of each virtual machine and CPU utilization of hypervisor node in real time. These data storage details are shared with a central database for calculating the performance metrics. This central database is deployed in OpenStack controller node. The data stored as the average megahertz consumed by each virtual machine during the last measurement of the time t. In practical CPU utilization of C(t1, t2) of the virtual machine function described below. Here t1 is the initial time and t2 is final time of the CPU utilization of the virtual machine. This CPU utilization of each node stored according to the time stamp.

$$\overline{C_i^v}(t_1,t_2) = \frac{n_i^v F(\tau_i^v(t_2) - \tau_i^v(t_1))}{t_2 - t_1}$$

CPU usage of $\overline{C_i^v}$ is VM classification (v) and VM number (i), which function of the interval between t_1, t_2 and n is a number of cores of the virtual machine F are the frequency of single-core virtual machine τ is CPU consumed by VM[2].

5.3.3 Live migration

Live migration of virtual machine is transferring the VM of one compute node to another compute node without downtime. This virtual machine live migration will impact on the system performance of the application. So, latency increased if a huge number of the VM placement operations are performing. William el at [29] explains the performance degradation of the application and downtime. The average estimated downtime expected 10% of the CPU utilization. The proposed algorithm should be performed minimum VM migration for eliminating SLA violation.

5.4 Design of OpenStack client wrapper

OpenStack has very broad supports in python client API. In this implantation, the wrapper supports migration and resource maintenance operations of the OpenStack environment. This wrapper classified into three stages.

1. Wrapper initialization
2. Decision-making algorithm
3. VM placement

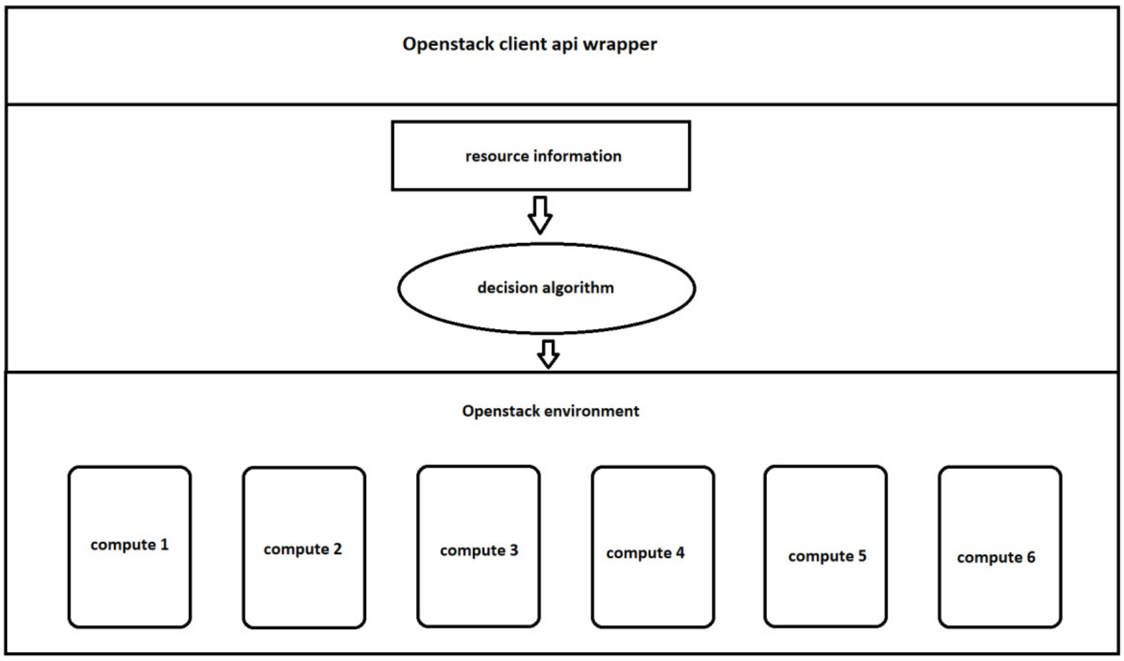

Figure 5.1 Overview diagram of an OpenStack implementation

1. Wrapper initialization:

This wrapper is mainly developed using OpenStack python API and numpy. To launch an instance in the OpenStack environment we need to pass the parameters like image name, volume name, key name, network id and compute availability zone. By default, OpenStack Nova scheduler will place instances in compute node using round-robin algorithm. So by using this wrapper before launching an instance in OpenStack environment perform internal calculations using Decision-making algorithm. This wrapper will provide all required resource information to Decision-making algorithm.

2. Decision-making algorithm:

By calculating the availability of the resources of each compute node this Decision-making algorithm provides the output of the compute node name where instance will launch. This part will be using First fit, Best Fit and Enhanced Best fit algorithms.

3. VM Placement

VM placement is performed in two types. These are

- Normal placement
- Threshold placement.

Normal placement will be performed whenever instance will be launched. Threshold values of each compute node will be set by the user (Max CPU utilization, memory usage, etc.). Whenever node utilization exceeds threshold value then script automatically trigger for performing live migration of the instances between compute nodes. It uses Bin packing algorithm for selecting the new host for the VM.

BIOS settings of each PM need to set according to Wake-on-LAN technology. So that we can control the compute node power settings. It is necessary to send magic packet using ether-wake program.

After VM placement operation this script automatically switches off the unused compute node using MAC address.

```
ether-wake <MAC address>
```

This wrapper is integrated with Grafana to display real-time resource utilization in the OpenStack environment.

5.5 OpenStack Monitor

OpenStack monitor is mainly developed for visualizing real-time cloud environment status. This tool is developed by using client OpenStack client API and Influx client. InfluxDB is time series database so, it can efficiently store all metrics using a time stamp.

The graphical interface of OpenStack monitor

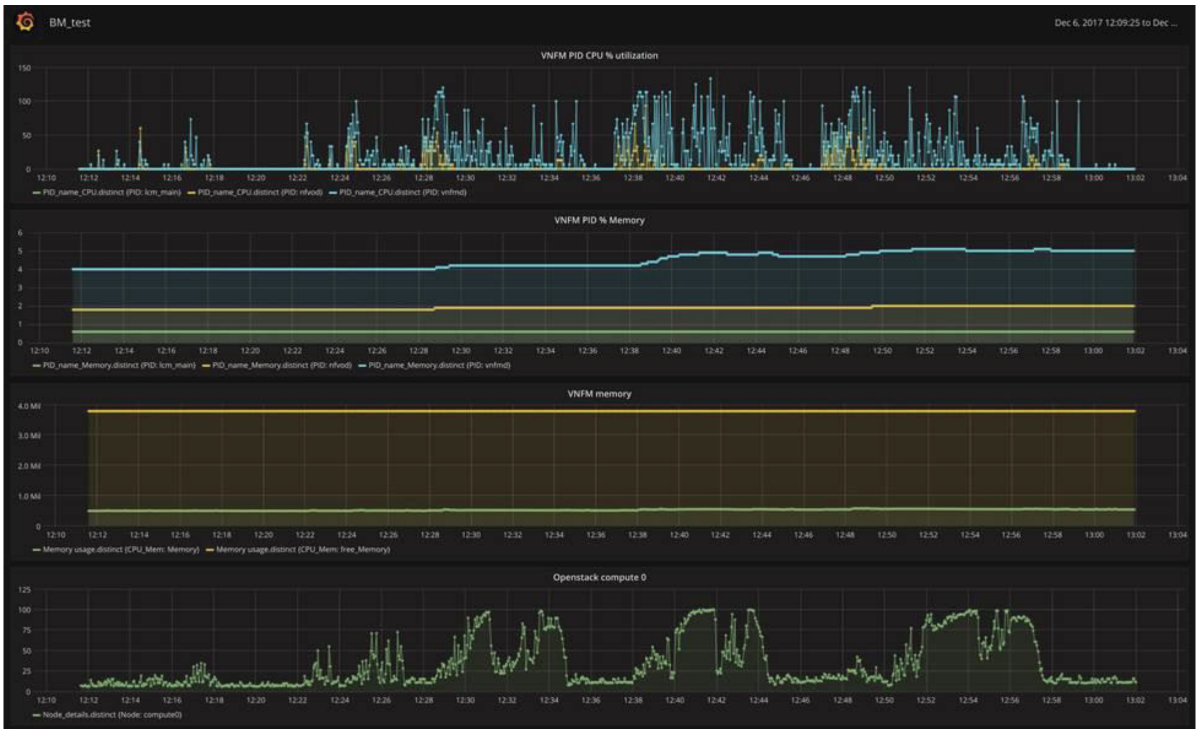

Figure 5.2 Visualization of Grafana dashboard -A

Main features of this OpenStack monitor are:

1. CPU utilization of the compute nodes
2. CPU utilization of controller nodes
3. No of VCPU deployed on the node
4. Disk space utilized in each node
5. Memory utilization of the node

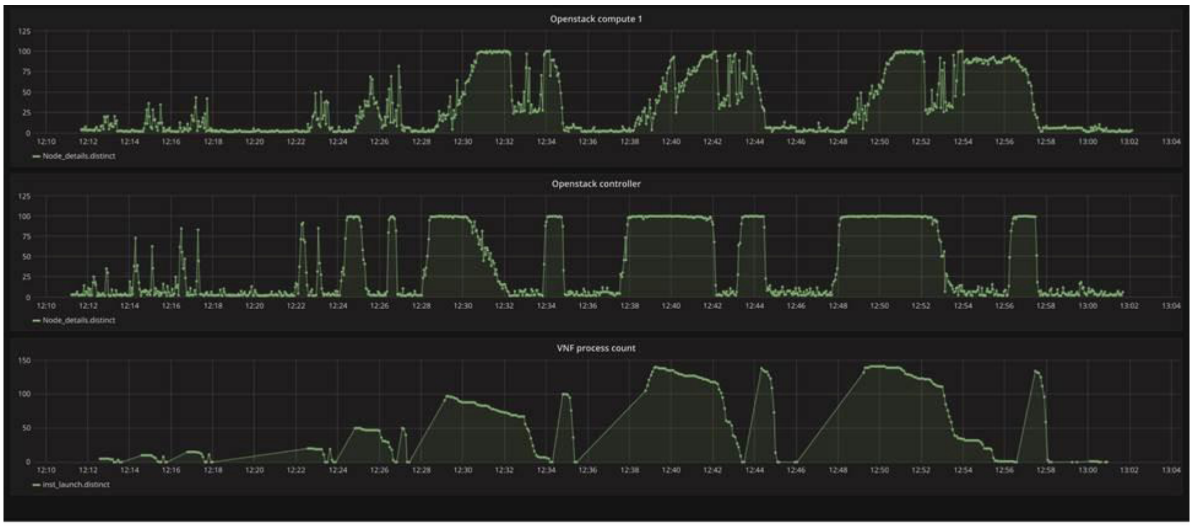

Figure 5.3 Visualization of Grafana dashboard -B

By knowing the information of all metrics, we know failure operation during testing. This tool dumps all metrics data from the database -2(compute node) to InfluxDb client.

5.2 Algorithms

Packing efficiency is always changed according to the variable bin sizes. So, the packing algorithm should be tested on variable size bins in short time period with more efficiency.

$$Packing\ Efficiency = \frac{sum\ of\ VM\ resources}{Used\ PM\ resources}$$

Description of bin packing algorithms in the following order.

1. First fit and First fit decreasing algorithm
2. Best fit decreasing algorithm
3. Enhanced best Fit algorithm

5.2.1 First fit algorithm

First fit algorithm is one of the oldest algorithms in memory management. This algorithm places the bins according to the first in first serve out (FIFO) basis. This algorithm is classified into two types.

a. First fit
b. First fit decreasing

In first fit decreasing algorithm initially, all bins are sort according to the bin sizes. So, most of the time it gives the better packing efficiency compared with the normal first-fit algorithm. Both algorithms are similar except sorting VM list in initial step So, below only mention first fit decreasing algorithm.

Algorithm 1: First fit decreasing algorithm

```
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3.     for host in Host list do
4.         if host has enough resources to VM then
5.             map_host ← Host
6.     if  map_host != NULL then
7.         add (map_host, VM) to map_to_VM
8.     return map_to_VM
```

5.2.2 Best fit decreasing algorithm:

In Best fit decreasing algorithm initially sort the VM resources and place the bins according to the power difference between after VM migration resources and before VM migration of host.

difference= (size of the host + VM) – (size of the host)

```
Algorithm 2: Best fit decreasing algorithm
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3.    for host in Host list do
4.       if (0 < host utilization > threshold (host, VM))
5.          power _difference ← power (host, VM) - power (host)
6.          If power _difference > max(power) then
7.             map_host ← host
8.             max(power) ← power _difference
9.    if map_host != NULL then
10.      add (map_host, VM) to map_to_VM
11.   return map_to_VM
```

5.2.3 Enhanced Best fit algorithm:

This algorithm is little different compared with the best-fit decreasing algorithm. In Best fit decreasing algorithm, we first sort the VM resources and place the bin according to the power difference between after VM migration resources and before VM migration of host. But in this approach, we are sorting both VM and Host each iteration.

Enhanced Best Fit algorithm

```
Algorithm 3: Enhanced Best fit algorithm
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3. Sort host list in order of decreasing CPU utilization
4.    for host in Host list do
5.       if (0 < host utilization > threshold (host, VM))
6.             map_host ← host
7.    if map_host ! = NULL then
8.       add (map_host, VM) to map_to_VM
9.   return map_to_VM
```

6 RESULTS

This section has to compare different parameters for the Best fit decreasing, enhanced best fit decreasing, first fit decreasing and first fit algorithm with Planet lab, google compute, Bitbrain and Raw dataset. So, we have performed the simulation 16 times to cover all our test cases.

Comparing metrics for placement algorithm:

1. Energy consumption
2. Number of VM migration
3. Number of host shutdown
4. Average migration time (Sec)
5. SLA violation

6.1 Energy Consumption

Energy consumption is depending on how many PM is using during operation. It is also changed according to the usage of the PM resources. From figure 6.1, it clearly shows that enhanced Best fit algorithm gives the better results in all of the test cases because we are sort the host resources and VM resources every iteration.

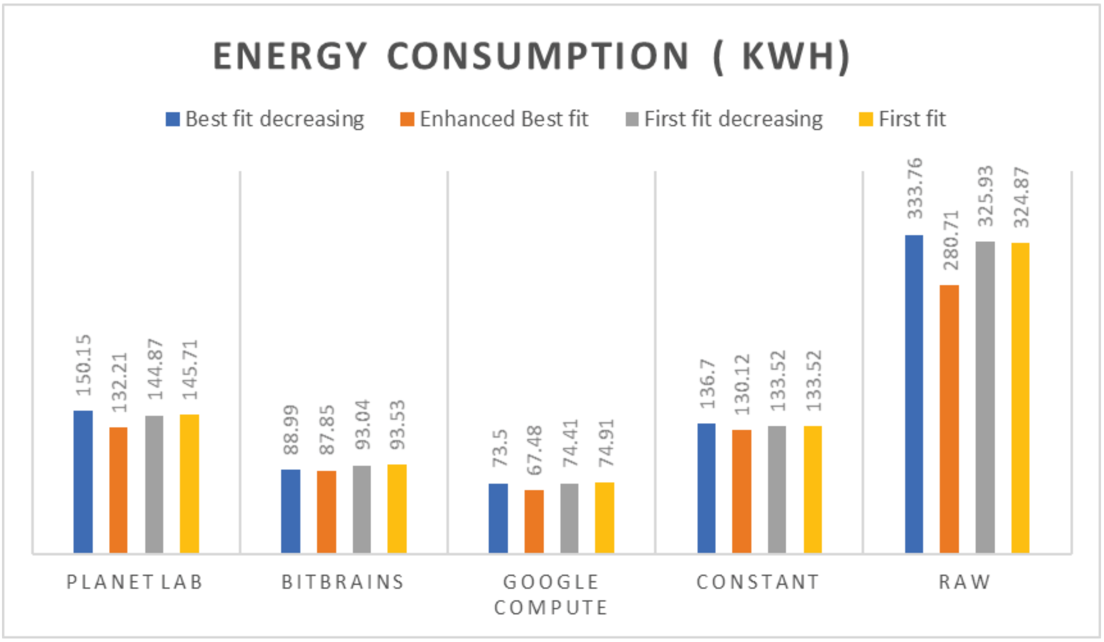

Figure 6.1 Comparison of energy consumption (kW h)

Below table represents the mean and confidence interval of each dataset.

Dataset	Mean	Std deviation	95% CI
Planet lab	143.24	7.71	135 to 151
Bitbrains	90.85	2.85	88.1 to 93.7
Google compute	72.58	3.45	69.2 to 76
Constant	133.47	2.69	130 to 136
Raw dataset	316.32	24.1	292 to 340

6.2 Number of VM migration

Less number of VM migrations means efficient consolidation, less traffic in cloud network and less SLA violation for VM migration. In this scenario, Enhanced best-fit algorithm gives better results compared with remaining algorithms except for constant database. In the constant dataset, best fit decreasing gives the better results.

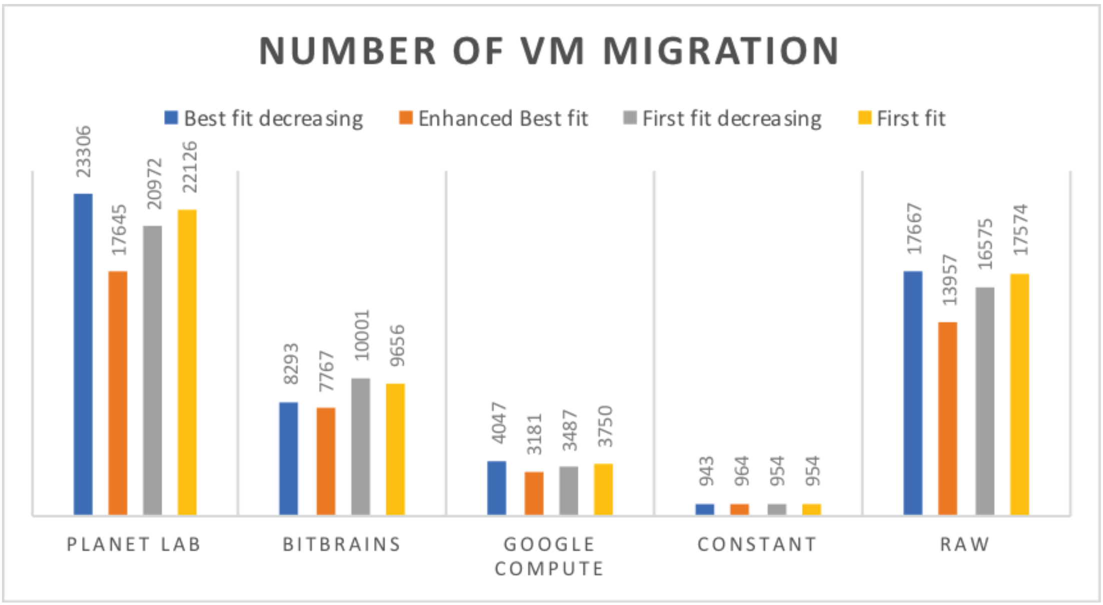

Figure 6.2 Comparison of the number of VM migration

Below table represents the mean and confidence interval of each data set.

Dataset	Mean	Std deviation	95% CI
Planet lab	21012.25	2440	18600 to 23400
Bitbrains	8929.25	1070	7880 to 9980
Google compute	3616.25	369	3260 to 3980
Constant	953.75	8.59	946 to 962
Raw dataset	16443.25	1730	14700 to 18100

6.3 Number of Hosts shut down

This case refers that number of time power OFF the trigger to PM. A minimal number of Host shutdown refers to the best efficient algorithm. From Figure 6.3, it is easy to observe that the number of host shutdown in the best fit enhances methods reduced significantly. That means when a host is put to sleep mode it stays there for a long time hence proves the efficiency of our algorithms.

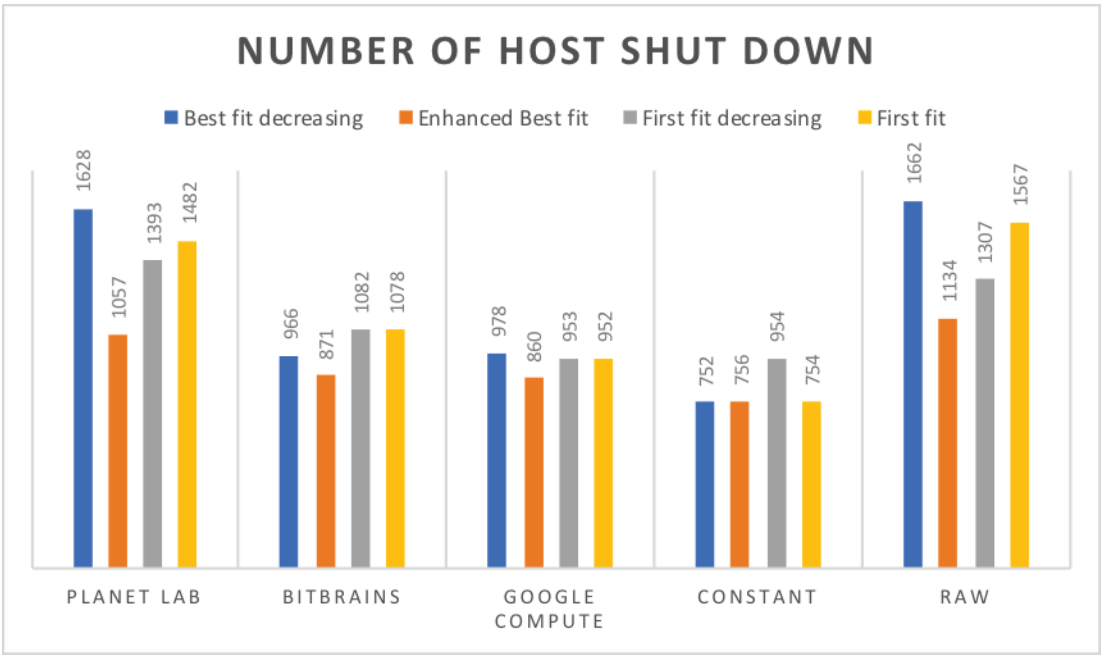

Figure 6.3 Comparison of the number of hosts shut down

In this scenario, Enhanced best-fit algorithm gives better results compared with remaining algorithms except for constant load database. In the constant dataset, first fit algorithm gives the better results.

6.4 Average SLA violation

This is one of the important metrics to measure the packing efficiency in PM. As discussed earlier minimal SLA violation will be stated as an efficient algorithm. So, In this simulation, we are using minimal migration time as preferred allocation policy in CloudSim. So, This section results look almost similar to other algorithms.

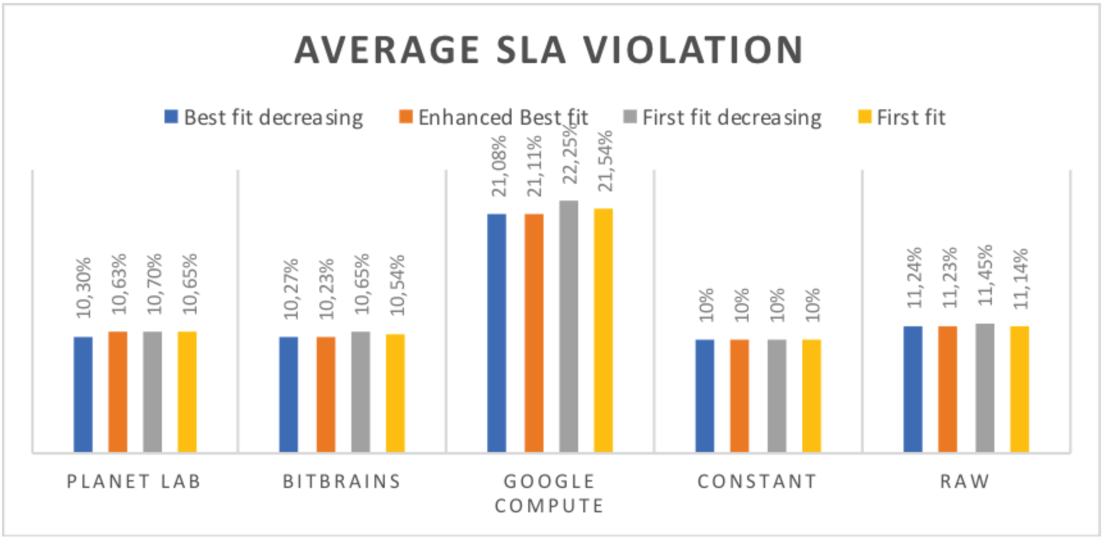

Figure 6.4 Comparison of average SLA violation

Below table represents the mean and confidence interval of each data set.

Dataset	Mean	Std deviation	95% CI
Planet lab	10.6	0.182	10.4 to 10.8
Bitbrains	10.4	0.205	10.2 to 10.6
Google compute	21.5	0.545	21 to 22
Constant	10	0	10 to 10
Raw dataset	11.3	0.131	11.2 to 11.4

6.5 Execution time (Sec)

Execution time is referred as simulation processing time to place all VMs. In this case, minimal execution time will be stated as the better approach. Because it completes place all VM migration operations in Datacenter. In this scenario Enhanced first fit decreasing algorithm gives better results compared with remaining algorithms.

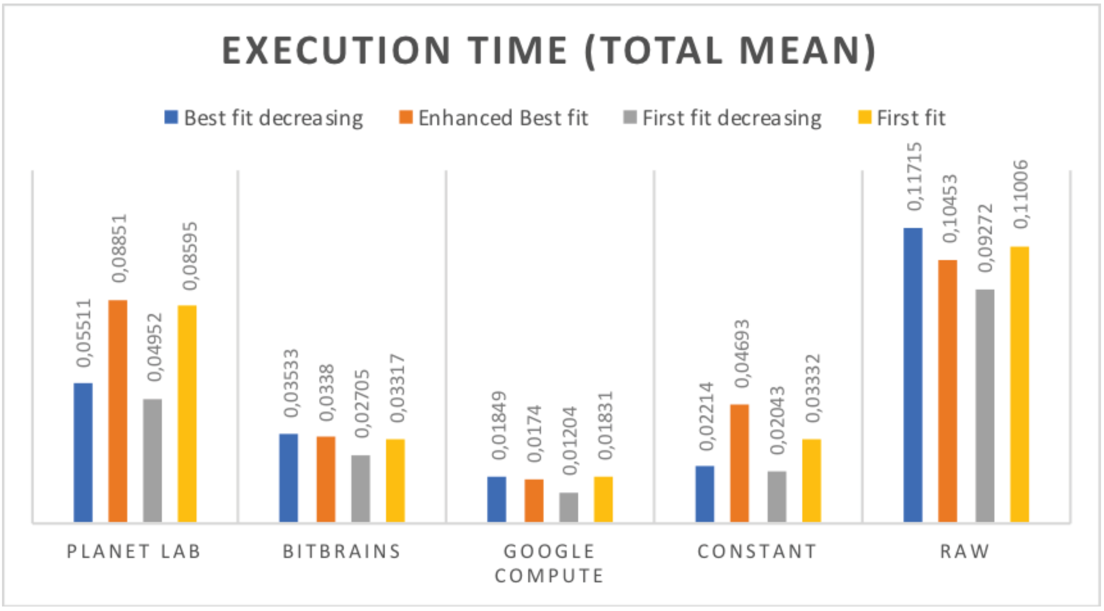

Figure 6.5 Comparison of Execution time (sec)

7 CONCLUSIONS

This research was done in order to find and evaluate and optimize the power consumption of the datacenters using Bin Packing algorithms. The outcome of the research comes as expected and energy consumption can be optimized using placement algorithms. CloudSim simulator was used for the simulation of the algorithms.

In above section, we are comparing different parameters with different algorithms. Most of the case enhanced best-fit algorithm gives the better results. In Execution time scenario First fit decreasing gives the better results compared with other algorithms.

For practical approach, OpenStack platform is used and we have Developed a python wrapper to integrate this model into the OpenStack environment using python client. Grafana dashboard used for monitoring real-time OpenStack resource information. the migration of the VMs operations is performed between the compute nodes. For a selection of the compute node enhanced Best fit algorithm was used.

The results proved that energy can be optimized in cloud data centers using Bin packing algorithms and performance of the physical machines are not degraded in a significant way.

8 QUESTIONS AND ANSWERS

1. **How can energy consumption be optimized by using bin packing placement algorithm?**

   The answer to this question is explained in section 5.1. Bin packing placement algorithm can be optimized by minimal degradation by taking care of low downtime and threshold values. In a practical approach, we are using OpenStack live migration technique so, VM can migrate one compute node to another compute node without any downtime. For this migration different algorithms were used. The algorithm is accepting the VM list and host list as an input gives the destination Host as an output. This algorithm will be packing the VMs limited PM and power off the unnecessary PM nodes.

2. **How should a heuristic model be defined to provide an efficient method to optimize the energy consumption in the data center?**

   For design a heuristic model we should consider the parameters like Energy consumption, migration time, number of VM migration, SLA violation as described in the result section. The answer to this question is explained in section 5.3. In this research, we have designed an experimental setup in the OpenStack environment. In wrapper script, we explain how the algorithm works and how power consumption will have minimized during the VM placement. We have taken input as an instance flavor and expected output will be the destination of compute node by calculating the resource availability of the PM. This wrapper script provided the additional option for performing bin placement to minimizing the entire data center.

3. **How is the overall Number of VM migrations of node affected when performing bin packing algorithm?**

   CPU utilization of the node is a very important parameter in VM migration. If CPU utilization of the host is exceeded, then degradation of the resource will be performed. Setup for the experiment is done in section 5.3 and implemented an OpenStack platform. The first user can set up the threshold values for each compute node. Load generator tool used for the generate the load for performing the Overload Operations and reduce the load by migration the VM. However, as discussed in results section Efficient bin packing algorithm performed less number of VM migrations rather than performing larger VM placement.

9 FUTURE WORK

This research mainly concentrated on bin packing problem in data centers. In future, by applying the machine learning it can give the improved solution to solve Bin-packing problem[30]. Research is needed for solving 3-D bin packing algorithm in a real-time environment. This model can be improved by using reinforcement learning algorithms.

Better to have a new simulation framework instead of CloudSim with a lot of new features like energy consumption of equipment. Design this simulator on Python so that it can easily integrate with other Opensource projects.

We assumed that the research will be carried out in a heterogeneous environment. However, implement this model in the large environment would be better for best results.

10 REFERENCES

[1] M. Mishra and A. Sahoo, “On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach,” in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 275–282.

[2] M. Mishra and U. Bellur, “Whither Tightness of Packing? The Case for Stable VM Placement,” *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, pp. 481–494, Oct. 2016.

[3] M. Forsman, A. Glad, L. Lundberg, and D. Ilie, “Algorithms for Automated Live Migration of Virtual Machines,” *Journal of Systems and Software*, vol. 101, pp. 110–126, 2015.

[4] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, “Performance and energy modeling for live migration of virtual machines,” *Cluster Comput*, vol. 16, no. 2, pp. 249–264, Jun. 2013.

[5] A. Beloglazov and R. Buyya, “Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.

[6] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers,” *Concurrency Computat.: Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[7] A. Beloglazov and R. Buyya, “OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds,” *Concurrency Computat.: Pract. Exper.*, vol. 27, no. 5, pp. 1310–1333, Apr. 2015.

[8] A. Andrae, “Total Consumer Power Consumption Forecast,” 05-Oct-2017.

[9] T. D. Burd and R. W. Brodersen, “Design issues for Dynamic Voltage Scaling,” in *Proceedings of the 2000 International Symposium on Low Power Electronics and Design, 2000. ISLPED ’00*, 2000, pp. 9–14.

[10] K. Ren, C. Wang, and Q. Wang, “Security Challenges for the Public Cloud,” *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[11] R. Jain and S. Paul, “Network virtualization and software defined networking for cloud computing: a survey,” *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[12] M. R. Ahmadi and D. Maleki, “Performance evaluation of server virtualization in data center applications,” in *2010 5th International Symposium on Telecommunications*, 2010, pp. 638–644.

[13] N. Jain and S. Choudhary, “Overview of virtualization in cloud computing,” in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1–4.

[14] L. Yan, “Development and application of desktop virtualization technology,” in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011, pp. 326–329.

[15] S. Abidi, S. Krichen, E. Alba, and J. M. Molina, “Improvement heuristic for solving the one-dimensional bin-packing problem,” in *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, 2013, pp. 1–5.

[16] “The CLOUDS Lab: Flagship Projects - Gridbus and Cloudbus.” [Online]. Available: http://www.cloudbus.org/cloudsim/. [Accessed: 10-May-2018].

[17] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms,” *Software Practice and Experience*, vol. 41, pp. 23–50, Jan. 2011.

[19] K. Sunil Rao and P. Santhi Thilagam, “Heuristics based server consolidation with residual resource defragmentation in cloud data centers,” *Future Generation Computer Systems*, vol. 50, no. Supplement C, pp. 87–98, Sep. 2015.

[20] M. Gabay and S. Zaourar, “Vector bin packing with heterogeneous bins: application to the machine reassignment problem,” *Annals of Operations Research*, vol. 242, Sep. 2015.

[21] H. Hallawi, J. Mehnen, and H. He, “Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation,” *Future Generation Computer Systems*, vol. 69, no. Supplement C, pp. 1–10, Apr. 2017.

[22] M. A. Kaaouache and S. Bouamama, “Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud,” *Procedia Computer Science*, vol. 60, no. Supplement C, pp. 1061–1069, Jan. 2015.

[23] Z. Tang, Y. Mo, K. Li, and K. Li, “Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment,” *J Supercomput*, vol. 70, no. 3, pp. 1279–1296, Dec. 2014.

[24] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, “Implementation and performance analysis of various VM placement strategies in CloudSim,” *Journal of Cloud Computing*, vol. 4, no. 1, Dec. 2015.

[25] J. Xu and J. A. B. Fortes, “Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments,” in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int’l Conference on Int’l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 179–188.

[26] D. Wilcox, A. McNabb, and K. Seppi, “Solving virtual machine packing with a Reordering Grouping Genetic Algorithm,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 362–369.

[27] J. Chen, K. Chiew, D. Ye, L. Zhu, and W. Chen, “AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines,” in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 235–242.

[28] “GWA-T-12 Bitbrains.” [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. [Accessed: 08-May-2018].

[29] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, “Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation,” in *Cloud Computing*, 2009, pp. 254–265.

[30] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, “Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method,” *arXiv:1708.05930 [cs]*, Aug. 2017.