Addis Ababa Institute of Technology
School of Electrical & Computer Engineering

Energy Efficient Virtual Machine Placement Algorithms in OpenStack Neat

By
Fikru Feleke

Advisor
Dr. Surafel Lemma

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

November 13, 2018

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical & Computer Engineering

Energy-aware Virtual Machine Placement Algorithms in OpenStack Neat

By
Fikru Feleke

Approved by:

Chairman, School Graduate Committee	Signature	Date
Dr. Surafel Lemma 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.

Fikru Feleke	_____	_____
Author	Signature	Date

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

Dr. Surafel Lemma	_____	_____
Advisor	Signature	Date

Abstract

Cloud computing provides a computing capability through the Internet. It enables organizations or individuals to have a computing power without deploying and maintaining their own Information Technology (IT) infrastructure. As a cloud is realized on a vast scale data-center, it consumes an enormous amount of energy. Several research have been conducted on consolidating Virtual Machines (VMs), logical computing machines that are hosted in servers, to minimize energy consumption. Among the proposed solutions OpenStack Neat is notable for its practicality. OpenStack Neat is an open-source VM consolidation framework that can seamlessly integrate to OpenStack, one of the most common and widely used open-source cloud management tool. The framework has components for deciding when to migrate VMs and selecting suitable hosts for the VMs (VM placement). The VM placement algorithm of OpenStack Neat is called Modified Best-Fit Decreasing (MBFD). MBFD is based on a heuristic that handles only minimizing the number of servers. The heuristic is not only less energy efficient but also increases Service Level Agreement (SLA) violation and consequently cause more VM migrations. To improve the energy efficiency, we propose VM placement algorithms based on both bin-packing heuristics and servers' power efficiency. In addition, we introduce a new bin-packing heuristic called a Medium-Fit (MF) to reduce SLA violation and VM migrations. To evaluate the performance of the proposed algorithms we have conducted experiments using CloudSim on three cloud data-center scenarios: homogeneous, heterogeneous and default. The workloads that run in the data-center scenarios are generated from traces of PlanetLab and Bitbrains clouds. The results of the experiment show up-to 67% improvement in energy consumption and up-to 78% and 46% reduction in SLA violation and amount of VM migrations, respectively. Moreover, all improvements are statistically significant with significance level of 0.01.

Keywords *virtual machine consolidation, virtual machine placement, bin packing, OpenStack, OpenStack Neat.*

Acknowledgment

I would like to express my gratitude to my advisor Dr. Surafel Lemma. His experienced and attentive supervision helped me to build a thesis out of otherwise pieces of unorganized immature ideas. The success of this thesis would be improbable without his excellent direction and proper follow up throughout the thesis progress.

My special thanks goes to AAU instructors specially to Dr. Mesfin Kifle and Dr. Beneyam Berehanu whose critical comments were effective for the quality as well as timely completion of the thesis.

I am very grateful to my family and my partner whose companionship and encouragement is immeasurable for the success my work. My gratitude also goes to the many friends for their motivation and support.

This research and the graduate program, Telecommunication Engineering, of which this research is part of is sponsored by **ethio telecom**.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AF Any-Fit.

APR Asymptotic worst-case Performance Ratio.

BF Best-Fit.

BFD Best-Fit Decreasing.

CPU Central Processing Unit.

FF First-Fit.

FFD First-Fit Decreasing.

IaaS Infrastructure as a Service.

IT Information Technology.

LR Local Regression.

MBFD Modified Best-Fit Decreasing.

MF Medium Fit.

MFD Medium Fit Decreasing.

MFPED Medium Fit Power Efficient Decreasing.

MHOD The Markov Overload Detection.

MIPS Millions of Instruction Per Second.

NF Next-Fit.

OTF Overload Time Fraction.

PaaS Platform as a Service.

PABFD Power Aware Best-Fit Decreasing.

PE Power Efficiency.

PEBFD Power Efficient Best Fit Decreasing.

PEFFD Power Efficient First Fit Decreasing.

RAM Random Access Memory.

SaaS Software as a Service.

SLA Service Level Agreement.

THR Averaging threshold-based.

VM Virtual Machine.

WF Worst-Fit.

Chapter 1

Introduction

According to Armbrust et al. [1], cloud computing refers to the applications delivered as services over the Internet. The hardware and systems software in the data-center that provide those services together constitute what we will call a cloud. Organizations offering the cloud in a pay-as-you-go manner are called cloud providers. Besides, organizations can deploy their own cloud computing hardware and software for private use. In general, a cloud has four deployment models: private, public, hybrid and community [2, 3].

Private The cloud infrastructure has been deployed, maintained and operated by a specific organization.

Public Clouds are owned and operated by a third-party cloud service provider, which deliver their computing resources like virtual machines, servers and storage over the Internet. This enables a consumer to develop and deploy a service in the cloud with very little financial outlay compared to the capital expenditure requirements associated with other deployment options.

Hybrid Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, hybrid cloud gives businesses greater flexibility and more deployment options.

Community Community cloud is a collaboration of infrastructure from multiple organizations with common interest. The cloud is managed by either participating organizations or a third party.

Cloud computing environment can be viewed as a layered architecture in which the lower layer provide the required resource to run the upper layer service as shown in Figure 1.1. The services offered by cloud computing practically fall into three broad categories [2]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service

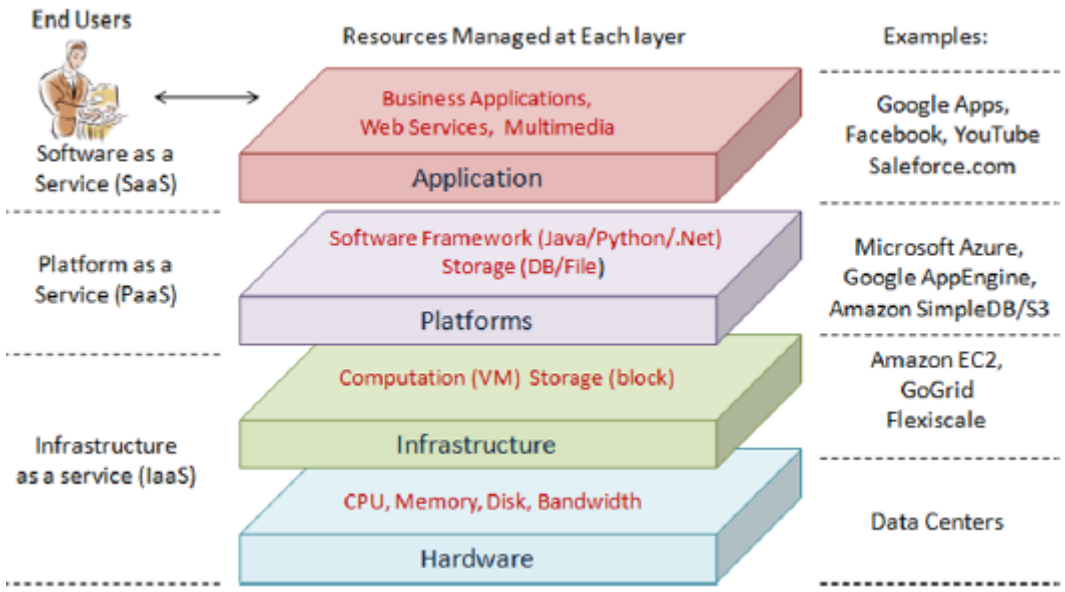


Figure 1.1: Cloud computing architecture [2].

(SaaS).

IaaS The most basic category of cloud computing services. With IaaS users rent Information Technology (IT) infrastructures on a pay-as-you-go basis. The infrastructures include servers, virtual machines, networking and storage.

PaaS Provides complete development and deployment environment in the cloud, with resources that enable to deliver cloud-based applications. Examples of PaaS include Google App Engine and Microsoft Windows Azure.

SaaS A service for delivering software applications over the Internet, on demand and typically on a subscription basis. With SaaS, cloud providers host and manage the software application and underlying infrastructure. A SaaS provider also handles maintenance like software upgrades and security patching. Examples of SaaS include Facebook and YouTube.

This thesis focus on efficient utilization of the lowest service layer, IaaS. IaaS is realized on large-scale, usually on distributed data-centers. Such infrastructure is known to consume an enormous amount of energy [4]. In 2012, energy consumption by data centers worldwide was 300 - 400 Tera-watt hour, about 2% of the global electricity usage and it is estimated to triple by 2020 [5]. Virtualization and consolidation are the two main technologies that enable cloud computing in general and IaaS in particular. Virtualization, by abstracting the hardware, creates logical resource groups called Virtual Machines (VMs). A VM has its own operating system and assigns computing resources to applications.

Consolidation uses live migration [6] to optimize power utilization by running VMs in a

few servers as possible and putting the rest in power saving mode or turning them off. Turning off servers or putting them in a sleep mode saves a large proportion of power as the study of Meisner et al. shows [7]. According to the study a typical HP blade server consumes 450W at peak load, 270W at idle state and 10.4W at sleep mode.

Energy efficiency has a trade-off with a Service Level Agreement (SLA) which is another concern of consolidation. From the cloud customer point of view, all that matters is the resource demand of their applications – SLA – to be fulfilled. However, energy efficient algorithms may overload some hosts to minimize the number of active hosts in the data-center. When a host is overloaded some of its VMs may not fulfill their resource demand; which will cause a violation of SLA. Thus any good consolidation algorithm should provide a well-balanced energy efficiency and SLA assurance.

The VM migration in consolidation increases the network overhead. This constitute the third aspect of consolidation: the amount of VM migrations [8]. The amount of VM migrations is affected when VMs are migrated to save energy, and also when there are overloaded hosts and some VMs have to be migrated from them to maintain SLA.

To address the above consolidation issues, a number of research works are conducted [8, 9, 10, 11, 12, 13]. Not all researches deal with all aspects of consolidation: the work in [12] deals particularly with energy minimizing aspect of consolidation while the works in [9, 10] address minimizing SLA violation as well. Still, some of the works [8, 11, 13] deal with all three aspects of consolidation but their work is not targeting cloud operating system that has the options or architecture to implement them. In the following section, we will investigate a consolidation technique that not only addresses all the aforementioned aspects but also is implementable by a widely used cloud computing tool called OpenStack.

1.1 Consolidation Framework for OpenStack

There are several cloud computing platforms for managing the infrastructure in a cloud. The list includes: Google Borg, Microsoft Apollo, Apache Mesos, Eucalyptus and OpenStack [14, 15]. OpenStack is a very widely used open-source tool for cloud infrastructure management and is supported by large community [15, 16]. Even though, OpenStack is vast and its components are rich in features, its scheduler does not directly support advanced optimization such as VM consolidation and load balancing. When a new VM request arrives to an OpenStack scheduler, the scheduler filters suitable hosts through a configured parameters such as available Central Processing Unit (CPU) or Random Access Memory (RAM). Those hosts are then prioritized by a weight function [17]. With those options an initial VM placement can be controlled by configuring or modifying filter

and weight functions. The limitation with the scheduler is that it has only an initial VM placement policy. It does not include a run-time resource optimization: that migrates VMs from overloaded hosts for maintaining SLA or underloaded hosts for reducing energy consumption.

In the next subsections we describe what has been attempted in literatures to resolve the problem of VM consolidation in OpenStack. We explore the proposed solutions for the VM consolidation framework, the VM consolidation algorithms and the integration of the framework with OpenStack.

VM consolidation frameworks

Huanel et al. [18] proposed an extension to OpenStack with a new resource scheduler framework. The framework supports an initial VM placement policy and an ongoing optimization policy per pool of servers. They have proposed a general purpose structure that can run user defined policy. The solution proposed has a sound conceptual architecture. However, the article does not include the detail of implementation and how it is integrated to OpenStack tool. Another work in regard to VM consolidation framework is that of Beloglazov and Buyya [19]. They have proposed an architecture and open-source implementation of a framework for dynamic VM consolidation in OpenStack cloud. Their framework called OpenStack Neat runs independently of the base OpenStack and applies consolidation processes by invoking public APIs of OpenStack.

VM consolidation algorithms

A practical VM consolidation framework constitutes algorithms that resolves three sub problems: (1) a decision when to start a VM migration (2) a selection of which VMs to migrate (3) a selection of hosts for placement [19, 20, 21]. Suitable algorithms have to be included for each category of sub problems. The VM consolidation algorithms of OpenStack Neat [19] are the following:

1. *A decision when to start VM migration.* VM migration process starts when there are hosts that are overloaded or underloaded. Some VMs are migrated from overloaded hosts to maintain SLA. From the underloaded host all VMS are migrated so that the host is turned off or put in power saving mode. Several heuristics are proposed for the host overload detection problem:

    – Averaging threshold-based (THR) overload detection algorithm: A static CPU utilization threshold is set above which a host is determined to be overloaded.

- Local Regression (LR) algorithm: Estimate the future CPU utilization using local regression.
- The Markov Overload Detection (MHOD) algorithm: Markov chain model is used to determine whether a host is normally serving a load or being overloaded.

Similarly for host underload detection the following heuristics are proposed:

- Average threshold based underload detection: A static CPU utilization threshold is set below which a host is determined to be underloaded.
- Minimum utilization: The minimum utilized host is decided to be underloaded.

2. *VM selection.* A VM selection decides which VMs to be migrated from overloaded hosts. The following are the proposed heuristics:
   - The minimum migration time policy: The minimum migration time policy migrates a VM that requires the minimum time to complete a migration relatively to the other VMs allocated to the host.
   - Random selection policy: Randomly selects VMs to be migrated.
   - Maximum correlation policy: VMs that have the highest correlation of the CPU utilization with the other VMs are selected to be migrated.
3. *VM Placement.* The VM placement is seen as a bin packing problem with variable bin sizes and prices, where bins represent the physical nodes; items are the VMs that have to be allocated; bin sizes are the available CPU capacities of the nodes; and prices correspond to the power consumption by the nodes. The Modified Best-Fit Decreasing (MBFD) heuristic is applied for the VM placement sub problem.

1.2 Statement of the Problem

An efficient VM placement algorithm in consolidation framework allocates resources in such a way that it satisfies VM's resource demand and minimize energy utilization with the least number of VM migrations possible. The problem of consolidation as a whole is well formulated in the works of Guazzone et al. (2012) [22] as an optimization function. The objective function is a linear combination of cost of energy, VM migration and performance degradation:

$$F(k) = w_e F_e(k) + w_m F_m(k) + w_p F_p(k) \tag{1.1}$$

Where,

- $F(k)$ is the value of the total cost during k^{th} round interval
- F_e is the energy consumption cost
- F_m is the VM migrations cost
- F_p is the performance degradation cost (cost of SLA violation)
- w_e, w_m, and w_p are nonnegative real values that are assigned to give different weights to the F_e, F_m, and F_p costs, respectively.

As described in Section 1.1, the three cost functions in (1.1) are not independent of each other. Migrating VMs to to save energy increase the amount of VM migrations and also some hosts may get overloaded (leading to SLA violation) due to the dynamic resource demand of the VMs. If we migrate some VMs from overloaded hosts to reduce SLA violation, obviously, it will increase the number of VM migrations. It may also increase the energy consumption if a new host need to be activated to place the migrated VMs.

The problem in (1.1) is a Mixed-Integer Nonlinear Program (MINLP) which is known to be NP-hard [8]. The only known solution for an MINLP is an exhaustive search and that is infeasible as the dynamical state in a cloud need fast converging solution. However, several approximation solutions are proposed in literatures [8, 23, 24, 25, 26]. In these approximate consolidation solution, the VM placement decision usually is handled with simple heuristics such as first-fit and best-fit decreasing.

OpenStack Neat provides an approximate consolidation framework for OpenStack cloud [23]. As stated in previous section OpenStack Neat applies a modification of Best-Fit Decreasing (BFD) called MBFD algorithm for VM placement. For the current VM to be allocated, MBFD selects an active host with a minimum available CPU that fits it. The host with smaller available RAM is favored to break a tie. The MBFD minimizes energy consumption by preferring an active server than turning on an inactive one. Moreover, as it is based on BFD it has high efficiency of packing VMs to the minimum number of hosts thus saving more energy. However, the algorithm has some weaknesses: (i) greedily selecting the most utilized host increase the overload probability and as a consequence more SLA violation and VM migration, and (ii) in a heterogeneous cloud it loses the benefit of favoring power-efficient servers. Indeed different host types have different consumption of power for the same load as shown Figure 1.2. The figure shows the power utilization of two host models – Hp ProLiant Ml110 G4 and Ibm X3250 Xeon X3470 – verses percentage of CPU utilization. Obviously, selecting a host that consumes a relatively lower power for a given load, when not all hosts necessarily have to run, will result in an overall saving of energy in the data-center. Thus the MBFD algorithm, by basing its selection

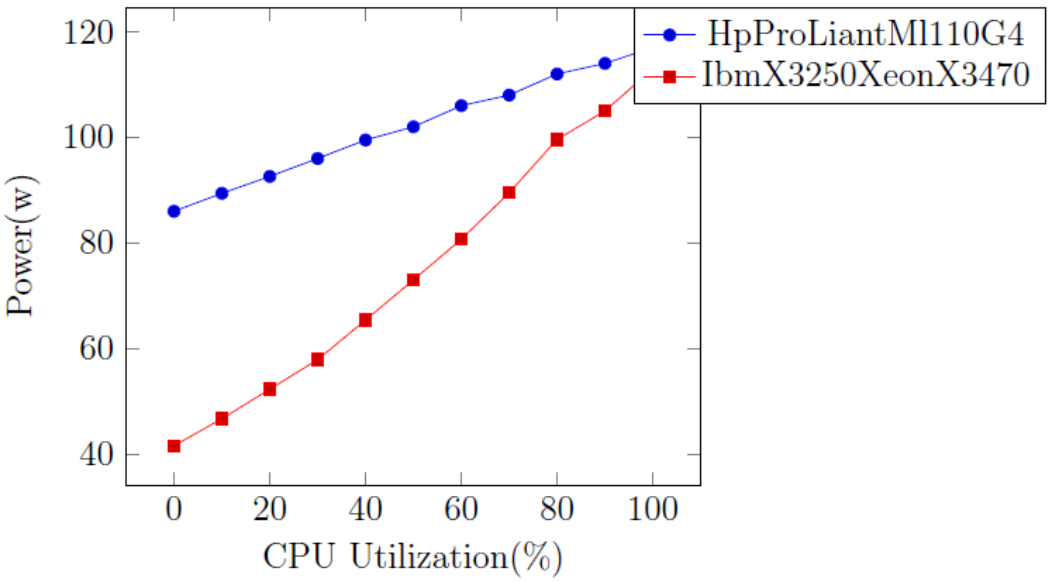


Figure 1.2: Power utilization specification for dual-core HP and quad-core IBM servers [27]. The CPU capacity of the HP server is 2 X 1800 MHz and that of IBM server is 4 X 2933 MHz.

criteria only on resource utilization level, misses power saving opportunity that would be attained by choosing lower power consuming hosts in case of a heterogeneous cloud. Summing up the above arguments, we draw the following *problem statement.*

The VM placement algorithm, MBFD, of OpenStack Neat framework has drawbacks with respect to exploiting better power saving opportunity, providing good SLA and minimizing VM migrations.

In this thesis, we try to propose a VM placement algorithms that minimize the aforementioned problems.

1.3 Objective of the Thesis

The objective of the thesis is to improve the VM placement algorithm of OpenStack Neat framework: to minimize energy consumption with reduced SLA violation and VM migration.

The following lists what we plan to do in order to achieve the objective.

- To investigate literature for VM placement techniques
- To propose VM placement algorithms for OpenStack Neat framework
- To simulate the proposed VM placement algorithms and cloud environments

- To evaluate the performance of proposed algorithms

1.4 Methodology

In this section we describe the methodology we plan to use to achieve the objective. It includes methods for investigation and proposing solutions, experimental design and the tools to be used for experiment.

Investigation of VM placement techniques

The mathematical foundation for cloud computing VM placement will be studied so as to select those heuristics that are likely to decrease energy consumption and SLA violation. Further, we examine how to implement those algorithms to OpenStack Neat framework and CloudSim, a toolkit for evaluation of resource provisioning algorithms [28]. We will also investigate VM placement algorithms evaluation techniques and data sources for simulation.

Proposing VM placement algorithms for OpenStack Neat framework

Based on the problem statement and our investigation of state of the art in VM placement techniques, we will propose new algorithms. The algorithms will be proposed such that they will likely improve the performance of energy utilization and SLA violation in OpenStack Neat framework.

Experimental design

The proposed solutions are evaluated using an experimental design called within-subjects design or repeated measures design [29]. In this experimental design there will be only one group of subjects that receive all treatments. The basic format of within-subjects design is shown in Table 1.1.

Table 1.1: Within-subjects design. The subject is one group of unit that undergoes two treatments. Observation is made for both treatments for comparison.

Subjects	*Time* $\rightarrow$	
Group 1	Tx_a	Obs_a
	Tx_b	Obs_b

In this research the subjects are cloud scenarios and are treated with two VM placement policies: one from baseline (Tx_a) and a second from proposed algorithms (Tx_b). Next, observations are taken for both treatments to make comparisons. The observations are metrics of energy consumption, SLA violation and number of VM migrations in the cloud. If the observation indicates a better metrics when the proposed algorithm (TX_b) is applied than is when baseline algorithm (Tx_a) is applied, we can reasonably conclude that the proposed solution brings performance improvement.

Tools

The experiment is planned to be conducted using CloudSim simulator which is widely used by researchers in industry and universities [28]. The workloads that run in data-center can be generated using real cloud traces. For data analysis, we plan to use software libraries from Matlab and Python.

1.5 Scope

Resources such as CPU, RAM, storage and so on affect energy utilization in a cloud. The problem we consider in this thesis is one dimensional; specifically considering CPU as a limiting resource. According to a survey by Zoltan Mann [30], CPU is considered the most critical resource in cloud computing. Moreover, CPU based power models work well for a wide variety of applications and platforms. As a result, many research works in consolidation algorithms, including OpenStack Neat framework, are based on CPU-power model [10, 11, 12, 19, 27].

1.6 Thesis Organization

The rest of the thesis are organized as follows. In Chapter 2, we lay the mathematical foundations used as a bases for proposing a VM placement algorithms. It introduces the Bin-packing problem and provides the packing efficiency of some common heuristics. In Chapter 3, we present the related works in the VM placement algorithms. Following it, in Chapter 4, we present the proposal of algorithms for the VM placement. We provide the proposed approach and the details of the proposed algorithms. Chapter 5 contains the experiment conducted and the results. The description of simulation tools, experiment setups, experiment datasets are given. The results of the experiment are analyzed and discussed. In the final chapter, Chapter 6, we draw some important conclusions, present

the contribution of the thesis work and state future research directions. The instructions for the baseline algorithms and some detail statistical tests are included as Appendix at the end of the document.

Chapter 2

Bin-Packing Algorithms for VM Placement

In the VM placement problem, one wants to assign VMs to hosts in such a way that the number of hosts is minimized. If we replace VMs by items and hosts by containers (bins) we will get the well known Bin-Packing problem. The mathematical foundations of Bin-packing algorithms were first studied by M.R. Garey and R.L. Graham in the early 1970s [31]. Soon a detailed analysis was presented by D.S. Johnson in his Ph.D. thesis [32]. The problems are well known to be NP-hard and hence an exact solution is unlikely to be found or is inefficient (e.g., for a very dynamic problem). However, many approximation heuristic algorithms are proposed in literatures. In this chapter we investigate and present the main results of the Bin-packing heuristics.

A survey of Bin-packing approximation algorithms has been conducted by Coffman et. al. in [31] and [33]. Their work comprehensively present worst-case and average-case performance analysis of different Bin-packing heuristics. In the following sections, we provide their main results in order to have an insight for proposing efficient VM placement algorithms. First, we present some notations and definitions. Then an asymptotic worst case and average case analysis of the classical Bin-packing heuristics: such as Next-Fit (NF), First-Fit (FF), Best-Fit (BF) and their variants are presented. In the final section we draw some important conclusions.

2.1 Notation and Definition

Problem Formulation

In classical one-dimensional Bin-packing problem one is given a sequence $L = (a_1, a_2, ..., a_n)$ of items, each with a size $s(a_i) \in (0, 1], 1 \leq i \leq n$, and then asked to pack them in a minimum number of unit capacity bins $B_1, B_2, ..., B_m$ [33]. If the bin size is c, (c!=1), it is scaled to unity and each item sizes are normalized by c accordingly. In some cases the item sizes are limited to interval $(0, \alpha]$, where $\alpha \in (0, 1]$.

For an algorithm A, let $A(L)$ be the number of bins taken by A to fill items in list L. The number of bins taken by an optimum algorithm is denoted by $OPT(L)$. The *optimum algorithm (OPT)* finds the fewest number of bins possible considering the different ways in which items can be placed.

Generally, the algorithms are divided into two categories: on-line and off-line algorithms. On-line algorithms are those that assign items as they arrive. Off-line algorithms assume that all items are known in advance and allow any rearrangement before assigning.

2.2 Worst-case Analysis for On-line Algorithms

The following definitions and theorems are from Coffman et al. [33, 31]. Let $R_A(L) \equiv A(L)/OPT(L)$ be the ratio of the number of bins taken by A to the minimum number taken by the optimum algorithm to pack list L, then the *absolute worst-case performance ratio* R_A for algorithm A is defined to be

$$R_A \equiv \inf\{r \geq 1 : R_A(L) \leq r, \text{ for all lists L } \} \tag{2.1}$$

Where,

- inf (short for infimum) means “greatest lower bound”
- r is real number

The *asymptotic worst-case ratio (or asymptotic performance ratio, APR)* is defined as:

$$R_A^{\infty} \equiv \inf\{r \geq 1 : \text{ for some } N > 0, R_A(L) \leq r \text{ for all lists L with } OPT(L) \geq N\} \tag{2.2}$$

The APR (R_A^{∞}) represents the worst-case performance ratio of the Bin-packing algorithm as the number N of bins gets infinitely large. In the next subsections the most common Bin-packing algorithms and their performance bounds are provided.

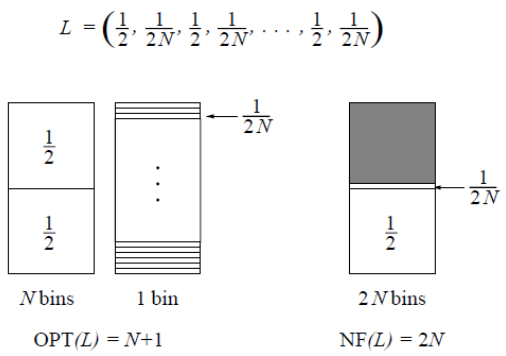


Figure 2.1: Bad list for Next-Fit (NF). The list L contains 2N number of items of size $\frac{1}{2}$ and the same number of items of size $\frac{1}{2N}$. By NF rule the first item of size $\frac{1}{2}$ is placed in the first bin and the second item with size $\frac{1}{2N}$ is placed in the same bin. The third item of size $\frac{1}{2}$ is placed in a new bin as there is no enough space in the first bin and so on. As a result, the NF rule will take 2N bins while there is a possible rule to pack the list in N+1 bins.

Next-Fit (NF)

After packing the first item, NF packs each successive item in the bin containing the last item to be packed, if it fits in that bin; if it does not fit, NF closes that bin and packs the current item in an empty bin.

Theorem 1 (Johnson et al. [34]) For next-fit one has

$$R_{NF}^{\infty}(\alpha) = \begin{cases} 2 & \text{if } \frac{1}{2} \leq \alpha \leq 1 \\ (1-\alpha)^{-1} & \text{if } 0 < \alpha < \frac{1}{2} \end{cases}$$

Theorem 1 gives an upper bound performance ratio of 2 for the NF algorithm. It also shows *items that are relatively smaller (smaller α) give better packing ratio than bigger ones* and this result also holds for other algorithms that follow. Example that illustrate the worst case performance ratio of NF is shown in Figure 2.1 where the list L contains 2N number of items of size $\frac{1}{2}$ and the same number of items of size $\frac{1}{2N}$. Items of size $\frac{1}{2}$ and $\frac{1}{2N}$ arrive alternatively. As the right side of the figure shows, the first item of size $\frac{1}{2}$ is placed in the first bin and the second item with size $\frac{1}{2N}$ is placed in the same bin. Then, there will not be enough space in the first bin for the third item of size $\frac{1}{2}$; and by NF rule the first bin is closed and a second bin is opened for the third item and the process repeats. Thus for the list shown NF will take 2N bins while the optimum with proper arrangement will take only N+1 bins. This asymptotically (as $n \to \infty$) gives the performance ratio of 2 (the worst-case performance for NF).

The time complexity of NF is $O(n)$ and as only one bin is ever open under NF, it is bounded space. The obvious disadvantage of NF is that it closes bins that could be used

for packing later items. An immediate improvement would seem to be never to close bins as we will see in the next algorithms.

Worst-Fit (WF)

If there is no open bin in which the current item fits, then WF packs the item in an empty bin. Otherwise, WF packs the current item into an open bin of smallest content in which it fits; if there is more than one such bin, WF chooses the lowest indexed one.

Although WF is not bounded space, its Asymptotic worst-case Performance Ratio (APR) is not better than NF, as the following theorem shows.

Theorem 2 (Johnson [32])

$$R_{WF}^{\infty}(\alpha) = R_{NF}^{\infty}(\alpha)$$

Theorem 2 shows that the next-fit and worst-fit have equal worst-case performance ratio for any list with item sizes less than or equal to α. The following two packing rules yield a better APR.

First-Fit and Best-Fit Algorithms

First-Fit (FF) FF packs the current item in the lowest indexed nonempty bin in which it fits, assuming there is such a bin. If no such bin exists, FF packs the current item in an empty bin [33, 31]. FF is neither linear time nor bounded space.

Best-Fit (BF) If there is no open bin in which the current item fits, then BF packs the item in an empty bin. Otherwise, BF packs the current item into an open bin of largest content in which it fits; if there is more than one such bin, BF chooses the lowest indexed one. This packing rule is a natural complement to WF which packs each item into a bin that has the maximum leftover space.

The time complexity of the algorithms (FF and BF) can be $O(n \log n)$ with appropriate data structure [33]. Theorem 3 states that the algorithms asymptotic worst-case ratio is 17/10 and improves as the size of items (α) gets smaller.

Theorem 3 (Johnson et al. [34])

$$R_{FF}^{\infty} = R_{BF}^{\infty} = \begin{cases} \frac{17}{10} & \text{if } \frac{1}{2} < \alpha \leq 1, \\ 1 + \lfloor \frac{1}{\alpha} \rfloor^{-1} & \text{if } 0 < \alpha \leq \frac{1}{2} \end{cases}$$

Any-Fit and Almost Any-Fit Constraints

The algorithms described so far FF, WF, and BF belong to a much larger class of an on-line heuristic satisfying the following condition [33, 31]:

Any-Fit constraint If $B_1, ..., B_j$ are the current nonempty bins, then the current item will not be packed into B_{j+1} unless it does not fit in any of the bins $B_1, ..., B_j$. The class of on-line heuristics satisfying the Any-Fit constraint will be denoted by AF.

Almost Any-Fit constraint An any-fit algorithm with the following constraint: it never packs an item into a partially filled bin with the lowest level unless there is more than one such bin or that bin is the only one that has enough room.

Both FF and BF do satisfy the Almost Any-Fit condition but not WF. The class of on-line algorithms satisfying Almost Any-Fit constraint will be denoted by AAF.

The following Theorem 4 shows that FF and WF are the best and the worst algorithms in AF, in the APR (Equation 2.2) sense.

Theorem 4 (Johnson [32]) For every algorithm $A \in AF$ and for every $\alpha \in (0, 1]$

$$R_{FF}^{\infty}(\alpha) \leq R_{A}^{\infty}(\alpha) \leq R_{WF}^{\infty}(\alpha)$$

The following theorem shows that any on-line algorithm $A \in AFF$ has the same APR as FF.

Theorem 5 (Johnson [32]) For every algorithm $A \in AFF$

$$R_{A}^{\infty}(\alpha) = R_{FF}^{\infty}(\alpha)$$

Semi-on-line Algorithms

Unlike the on-line algorithms, in semi-on-line algorithm more information is assumed about elements in the list. As a result better performance ratio is expected. In semi-on-line algorithms the following rules are allowed:

- Repack elements
- Look ahead to some later elements before assigning the current one
- Assume some preordering of the elements

Among the many results reported by Coffman et al. [31, 33], the behavior of heuristics with presorting of the lists is notable. If items are sorted in increasing size, their asymptotic performance ratio will not be lower than 1.540. An example is shown for FF

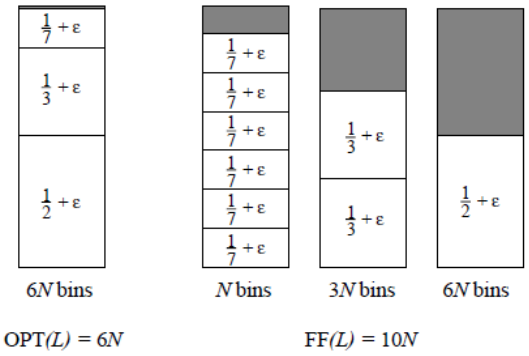


Figure 2.2: Bad list for First-Fit algorithm when items are arriving in increasing order of size. There are 6N number of $\{1/7+\epsilon, 1/3+\epsilon, 1/2+\epsilon\}$ sized elements where ϵ is a negligible real number. The optimum algorithm can pack them in 6N bins, one bin containing one item from each type. Using FF rule, the first 6N of $1/7+\epsilon$ sized items will be packed in N bins (each bin containing 6 items) and no space is left for other sized items; similarly, 6N of $1/3+\epsilon$ sized items will be packed to 3N bins (each containing two items); the last 6N of $1/2+\epsilon$ will be packed to 6N bins.

in Figure 2.2 which shows that when items are arriving in increasing order of size the performance ratio can be as worse as 1.67.

In lists where items arrive in decreasing order, smaller sizes fill-up the gaps left by large ones. If the list is presorted by decreasing item size, then R_A^∞ (Equation 2.2) is between 11/9 and 5/4 for any algorithm A $\in$ Any-Fit (AF).

2.3 Worst-case Analysis of Off-line Algorithms

An off-line algorithm has all items available for preprocessing, reordering, grouping, etc. before packing. It has been seen that most of the classical on-line algorithms achieve their worst-case performance ratio when the items are packed in increasing order of size (see for example Figure 2.2), or if small and large items are merged (see for example Figure 2.1). Thus, one is led to expect improved behavior by sorting of the items in decreasing order of size [31].

Packing a list sorted in decreasing order according to first-fit or best-fit rules gives the algorithms First-Fit Decreasing (FFD) and Best-Fit Decreasing (BFD), with much better asymptotic worst-case ratio.

Theorem 6 (Johnson et al. [34], Yue [35])

$$R_{FFD}^\infty = R_{BFD}^\infty = 11/9$$

For both first-fit and best-fit decreasing algorithms, the following absolute worst-case performance ratio holds

$$R_{FFD} = R_{BFD} = 1.5 \text{ [31]}$$

Even though BFD and FFD perform equal in worst-case, there is no clear winner in general sense. Examples where the performance ratio of BFD can be better or worse compared with that of FFD are given by Johnson in his PhD thesis [32].

2.4 Average-case Analysis

Average-case analysis deals with the study of asymptotic expected ratio of the algorithms. Some general average-case results hold for arbitrary distributions of item sizes but most of the specific expected values that have been computed concern continuous uniform or discrete uniform distributions, U[0, 1] [33].

Notation and Definition for Average-case Analysis

Let F denote an item-size distribution and let us denote its mean and variance by μ and σ^2. L_n denotes a list of n randomly generated items.

In the context of average case, many of the standard functions of lists become random variables: $s(L_n)$ (the sum of the item sizes), $OPT(L_n)$ (optimum number of bins for list L_n), $A(L_n)$ (number of bins under a bin packing algorithm, A for list L_n), and $R_A(L_n) = A(L_n)/OPT(L_n)$ () (performance ratio of A for list L_n). There is also interest in the derived random variable $W_A(L_n) = A(L_n) - s(L_n)$, which is the amount of wasted space in the bins of the packing of L_n by algorithm A.

The expected performance ratio and expected wasted space are defined as follows.

$$\bar{R}_A^n(F) \equiv E(R_A(L_n)) = E(\frac{A(L_n)}{OPT(L_n)})$$

$$\bar{W}_A^n(F) \equiv E(A(L_n) - s(L_n))$$

Where E is an expectation function. The asymptotic expected ratio for A under F is defined to be:

$$\bar{R}_A^\infty(F) \equiv \lim_{n\to\infty} \bar{R}_A^n(F)$$

Theorem 7 Next-Fit asymptotic expected ratio

$$\bar{R}^{\infty}_{NF}(U(0,1]) = 4/3$$

Theorem 8 Best-Fit and First-Fit asymptotic expected ratio

$$\bar{R}^{\infty}_{BF}(U(0,1]) = \bar{R}^{\infty}_{FF}(U(0,1]) = 1$$

How fast do the algorithms converge to the above expected ratio? The following two theorems shows the performance of expected wasted space (W) in packing of n items.

Theorem 9 For First-Fit we have

$$\bar{W}^{\infty}_{FF}(U(0,1]) = \Theta(n^{2/3})$$

As shown in the theorem 10, the algorithm Best-Fit turns out to be even better.

Theorem 10 For Best-Fit one has

$$\bar{W}^{\infty}_{BF}(U(0,1]) = \Theta(\sqrt{n}log^{3/4}n)$$

Note that the wasted space ratio ($\bar{W}/n$) get diminished for large n. For example for $n = 1000$, $\bar{W}/n = 0.1c1$ for FF and $\bar{W}/n = 0.072c2$ for BF; where c1 and c2 are constants that do not grow with n. For $n = 10000$, the values for FF and BF are $0.00046c1$ and $0.00028c2$, respectively.

2.5 Conclusion

In this chapter, we have investigated the Bin-packing algorithms. For the most common heuristics, we have presented their performance in worst-case and average-case ratios. We conclude the chapter with some important points which will be used for proposing energy efficient VM placement algorithms.

From the analysis of our investigation we can conclude that algorithms that satisfy one or more of the following conditions will have higher packing efficiency (better average and worst-case performance ratio):

- If the algorithm is based on First-Fit, Best-Fit or satisfy the Almost Any-Fit constraint (see performance results in Theorem 3, 5 and Section 2.4)
- If items are sorted by decreasing size (from on-line and semi-on-line algorithms performance analysis of Sections 2.2 and 2.3)

- If items have smaller sizes relative to their bin sizes (see for example Theorem 1 and 3)

The First-Fit Decreasing (FFD) and Best-Fit Decreasing (BFD) are the most common and highly efficient Bin-packing heuristics that satisfy the above two conditions. As a result, VM placement algorithms based on FFD and BFD are expected to minimize the number of servers required to place the VMs in a cloud. In the next chapter we will develop such algorithms for energy efficient VM placement.

Chapter 3

Related Works

As recalled from Chapter 1, energy efficient operation is one of the main concerns of cloud computing. Quite a lot of literatures have proposed energy aware cloud operation using VM consolidation technology [8, 9, 10, 11, 12, 13, 20, 24, 25, 26]. Energy efficient VM placement which is the concern of this thesis is part of the consolidation process. VM placement in consolidation process deal with the placement of VMs to hosts such that energy consumption of the cloud is minimized and the SLA of VMs is not degraded. In the next section, we present literature review of consolidation solutions and related works on VM placement algorithms.

3.1 Consolidation Solutions

In [10], Berral et al. modeled a cloud as a set of jobs or tasks to be distributed along a set of resources. They defined an optimization function as the profit of executing jobs with specified SLA minus the cost of power consumption. An exact solution to the optimum function required several minutes to schedule, even 10 jobs among 40 candidate hosts. So the first-fit and best-fit heuristics are proposed as an approximate solution. The best-fit solution has a result close to the optimal in extremely low time. The jobs considered for the experiment are Web-services and their resource usage are learned using a machine learning technique. In their later work, Berral et al. improved the optimum function by including a penalty for VM migration [11].

Multiple level grouping genetic algorithm is proposed to reduce carbon footprint in distributed cloud in works of Moghaddam et al. [12]. A similar model called segmented two level group genetic algorithm is used to formulate a unified server and data-center level optimization to minimize delay, the overall power, and physical resource cost for virtual desktop placement by Zhang et al. [13]. These works have scaled up the usual server

level modeling of genetic algorithm to a higher level such as data-center.

The problem of consolidation is well formulated in the works of Guazzone et al. as an optimization function [8, 22]. The objective function is a linear combination of cost of energy, VM migration and cost of performance degradation. The resulting mathematical programming is a Mixed-Integer Nonlinear Program (MINLP) which is known to be NP-hard. The only known solution to an NP-hard problem is an exhaustive search which is infeasible for dynamic cloud environment owing to its slow convergence. Thus, they resorted to an approximate solution computed by means of a Best-Fit Decreasing (BFD) strategy, which attempts to place the largest number of VMs on the fewest number of physical machines, while still preserving SLA constraints.

Farahnakian et al. proposed Utilization Prediction Aware Best-Fit Decreasing (UP-BFD) consolidation solution [26]. The UP-BFD enables proactive consolidation of VMs using a resource utilization prediction model. The model uses the K-nearest neighbor regression to predict CPU utilization of VMs and hosts based on the historical data.

The lack of consolidation scheduler in OpenStack tool have motivated Beloglazov and Buyya [19] to propose a VM consolidation framework and an open-source implementation for OpenStack cloud. The framework has host overload and underload detection, a VM selection and a VM placement part.

3.2 VM Placement Algorithms

In many of the approximate solutions for consolidation, the VM placement part is handled with simple heuristics such as a modified form of best-fit and first-fit decreasing [8, 19, 24, 25, 26]. In the works of Beloglazov et al., the VM placement problem is handled by the MBFD algorithm [19]. The algorithm deals with minimizing the number of active servers and is based on the bin-packing heuristic, BFD. The default VM placement algorithm in CloudSim cloud simulator is the Power-Aware Best-Fit Decreasing (PABFD) [20]. The PABFD places the current VM on a host that fits it and the estimated increase in power is the minimum. We see that in the context of PABFD the best-fit stands for the best in power-utilization for the VM to be placed rather than its mathematical definition – to place to the fullest host that fits the VM (see the definition of best-fit in Section 2.2). Chowdhury et al., proposed the Power-Aware Worst-Fit Decreasing (PAWFD) algorithm which favors a host with estimated increase in power utilization is the maximum (quite the opposite of PABFD) [24]. Their experiment has shown that PAWFD has better performance than their baseline algorithm, PABFD.

A comprehensive performance analysis of various VM placement algorithms is conducted

by Z. Mann and M. Szabo [25]. For overload and underload detection, the authors reuse algorithms from OpenStack Neat framework. The VM placement algorithms considered for comparison include PABFD and PAWFD. The best performing algorithms are the "Guazzone" [8] and the "Shi-AC" [36] algorithms. The "Guazzone" algorithm is from the works of Guazzone et al. [8] and it applies three host selection criteria: (i) powered-on host proceeds powered-off host, (ii) within powered-on or powered-off host category, hosts are selected by decreasing size of free CPU, and (iii) in case of same CPU capacity, hosts are selected by increasing values of idle power consumption. The "Shi-AC" is from Shi et al. and assigns a VM placement by favoring a server with the largest absolute CPU capacity [36].

To address the issue of high SLA violation and VM migrations caused by heuristics that only deal with minimizing the number of servers, Farahnakian et al. proposed prediction aware VM placement [26]. The proposed algorithm called Utilization Prediction Aware Best-Fit Decreasing algorithm (UP-BFD) chooses a host based on the prediction of future resource utilization. Their results show that UP-BFD performs better than BFD and FFD with no utilization prediction functionality.

In this thesis, instead of proposing a novel framework for VM consolidation, we built our work on existing OpenStack Neat framework and specifically improve the VM placement component. The VM placement algorithms we propose, like most of the above works, are based on bin-packing heuristics. However, as we reason out in the problem statement (Section 1.2), minimizing the number of servers by using heuristics like best-fit is not enough. The power utilization differences among hosts has to be also considered. Though, the PABFD in [20] looks at power differences among hosts, it does so for the current load (VM) only. It does not consider the idle power of hosts as well; which contribute significant part of the server's power utilization [7]. This research propose VM placement algorithms by modifying the bin-packing heuristics with the power-efficiency parameter. The power-efficiency, defined in (4.1), not only include power due to a load but also the idle power of hosts. Moreover, as argued in Section 1.2 and also learned from [26], best-fit based algorithms increase SLA violation and number of VM migrations. To lower SLA violation and number of VM migrations, we defined a new bin-packing rule called medium-fit. The medium-fit heuristic, when modified by power-efficiency, is efficient in reducing energy consumption, SLA violation and amount of VM migrations.

Chapter 4

Proposed VM Placement Algorithms

In the previous chapter, we have investigated Bin-packing algorithms that can be implemented to minimize the number of hosts required to handle a given workload of VMs. In this chapter, we first analyze factors affecting energy utilization in cloud computing environment. We then propose VM placement algorithms by modifying the Bin-packing heuristics.

4.1 Energy Utilization Factors

Given that a cloud has to serve a certain workload of VMs, it is obvious that as the load increases there will be an increase in energy consumption. Besides the load, there are other factors that affect the total energy: the number of active hosts that serve the workload and the power model of hosts running the load. The number of active hosts and power models of active hosts can be affected by the choice of VM placement algorithm. In Figure 4.1, we summarize energy utilization factors in cloud computing.

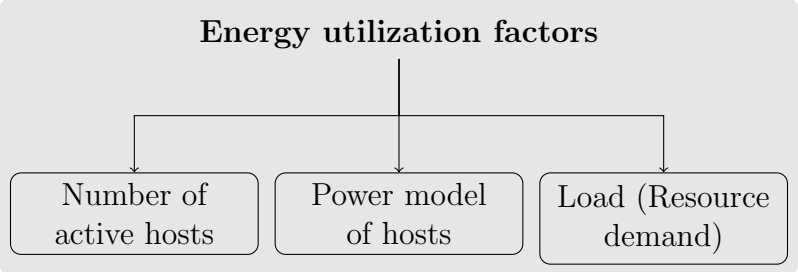


Figure 4.1: Illustration of energy utilization factors

The description of each factor including the way they affect energy consumption and possible mitigation are given below.

Load (Resource demand) The relation between CPU utilization and power is monotonic and mostly linear [30]. Which means that total energy in a data-center increases with the workload. It would be counter productive to sacrifice some VMs' workload resource demand to save energy, as there will be penalty due to SLA violation. However, the VMs can be assigned to a smaller number of hosts instead of distributing them to large number of hosts in order to reduce energy consumption.

Number of active hosts When a workload is assigned to a smaller number of hosts and the rest are either turned off or put in sleep mode, the total data-center energy can be minimized. This happens because the idle power (power at zero loads) contributes a large portion of the total power consumed; according to [37] a host can consume up to 70% of its peak power in the idle state. As investigated in Chapter 3, Bin-packing heuristics such as BFD and FFD can be adopted to minimize the number of hosts that handle the whole workload.

Power model of hosts Some hosts are more power efficient than others. For example, assigning loads to a quad-core host will usually be more power efficient than a dual-core host. That is so because adding a core can take a negligible amount of power compared with turning on or activating additional host [10].

To incorporate power-model of servers into our solution, we define Power Efficiency (PE) of a host as follows:

$$PE \equiv CPU_{total}/Power_{max} \tag{4.1}$$

Where,

- CPU_{total} is the CPU capacity of a host
- $Power_{max}$ is the power utilization of a host at 100 % CPU utilization. It is the sum of the idle power and power due to the maximum load.

The definition is in-line with the common sense definition of *efficiency* which is the ratio of output to input parameter. The efficiency becomes higher as the output gets higher and input gets lower. The same definition has been used to define energy efficiency of the "Lago" algorithm in [25].

4.2 Power-efficient Modified Heuristics

Previously we have analyzed the factors that affect energy consumption in cloud computing. Accordingly the following heuristics, illustrated in Figure 4.2, are expected to lead to an energy efficient VM placement algorithms:

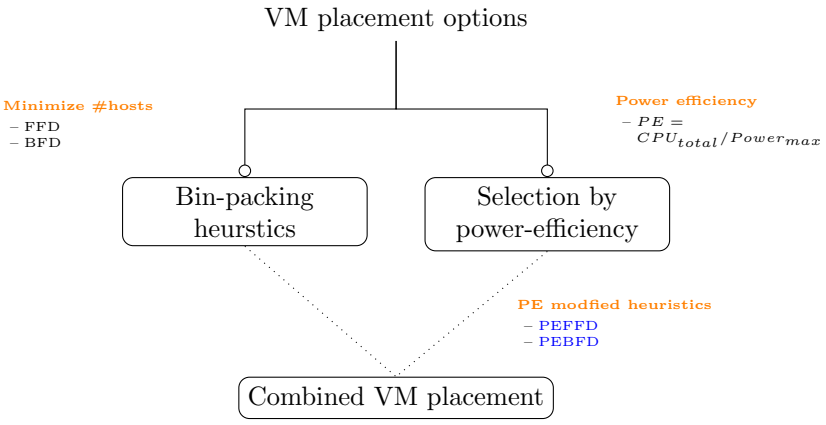


Figure 4.2: **Illustration of power-efficient modified heuristics.** The power-efficient modified heuristics modify the bin-packing rules by power-efficiency (PE) parameter.

1. **Minimize the number of active hosts** using Bin-packing heuristics such as FFD, BFD etc.
2. Favor hosts with **higher power efficiency (PE)**.
3. Using an algorithm that combines the above rules.

In this research we will implement the combined method (option 3) for better energy efficiency. The combination is done by modifying the bin-packing rules by power-efficiency (PE) and the details are given in Section 4.4.

4.3 Medium Fit Decreasing (MFD) Heuristic

The efficiency of practical VM placement algorithm not only has to be measured by its energy saving capability but also by its ability to avoid side effects: cost due to SLA violation and network overhead due to VM migrations. The best-fit based algorithms have a tendency to create overloading of some hosts when minimizing the number of active hosts and consequently more VM migrations by overload detection algorithm. For this reason, we proposed a Bin-packing algorithm with the intention of providing a good balance between minimizing energy and reducing overloading effect. We call the new heuristic Medium Fit (MF). We denote the version where items are sorted in decreasing order by the Medium Fit Decreasing (MFD).

We define the Medium-Fit algorithm as follows: Let L_D be a desired resource utilization level between overload-threshold and under-load threshold; say it is an average level between overload threshold and under-load threshold: $L_D \equiv (overload_{thr} + underload_{thr})/2$. Then the MF rule is defined to favor a host whose resource level has a minimum distance

from L_D. More precisely,

$$\text{Allocated-host} \equiv \arg\min_h |L_h - L_D|, \text{ where } L_h \text{ is utilization level of host h.} \quad (4.2)$$

If $L_D = overload_{thr}$, we have the best-fit algorithm. If on the other hand $L_D = underload_{thr}$ then it will be equivalent to the worst-fit. Hence the name medium-fit. For example if $overload_{thr}$ is assumed to be 0.9 (90%) and $underload_{thr}$ is assumed to be 0.3 (30%) then $L_D = 0.6$.

The reason why the MFD algorithm minimizes overload probability and at the same time minimize the number of active servers is explained in the following. Suppose that all hosts are below L_D and the underload detection algorithm selects the lowest loaded host for its VMs to be migrated. Then, the MFD algorithm takes each VM in turn and allocates it to the highest loaded host according to Equation (4.2). The process is repeated – taking VMs from the lowest loaded host to highest ones – until some hosts pass the desired level, L_D. The hosts whose VMs are migrated from are then turned off or put in sleep mode. This is minimizing the number of active servers without causing the highest loaded hosts pass overload-threshold. On the other hand, if some hosts are above the desired level, say by the long run underload migration process, then Equation (4.2) imply that any new VM migration (from underloaded or overloaded hosts) will be allocated to a host whose load level is near L_D. In both cases, MFD minimizes the overload probability, and as a consequence reduce SLA violation and VM migrations.

4.4 Proposed Algorithms

Before defining the proposed algorithms we present a generic flowchart for illustrating the basic idea. The generic flowchart contains the basic common blocks out of which the rest of algorithms are constructed. The generic flowchart is given in Figure 4.3.

In the generic flowchart (Figure 4.3) the algorithm first takes as input the VM lists to be migrated and host lists in the data-center. Then, for each VM sorted by decreasing resource utilization, the algorithm finds a host that fits the VM and is the best in terms of the particular rule of the algorithm. A host fits for a VM if it has enough resource for the VM. The best host is then determined by iteratively selecting the better host. Next, a VM and its best host are added to a vmPlacement map. When all VMs are exhausted, the vmPlacement, a map of VMs and allocated hosts, is returned.

The criteria for selecting whether a host is better-than another host is different between the VM placement algorithms. For example in MBFD a host is better-than another host if its CPU utilization level is higher than the other host. In the following sections we will

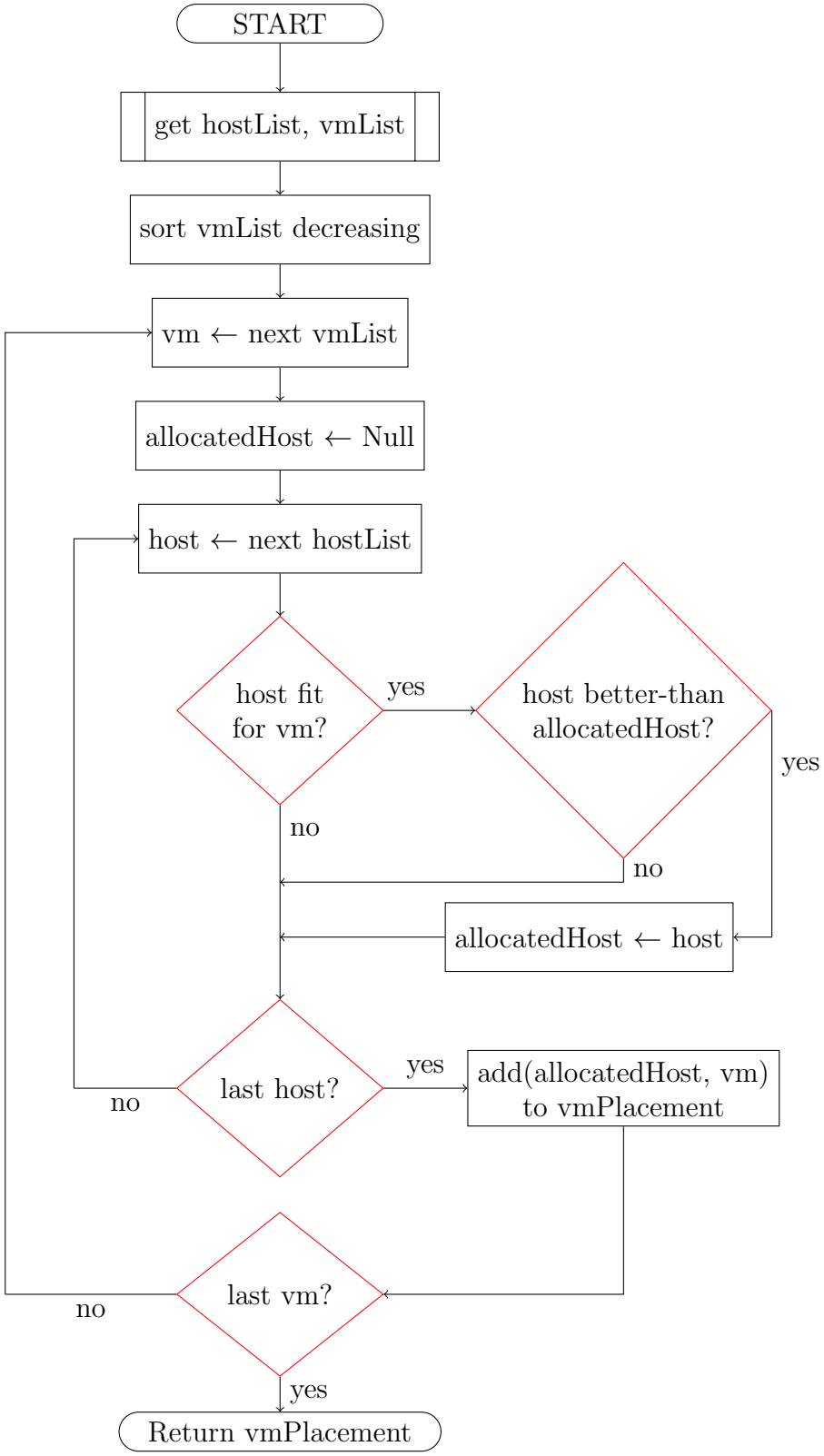


Figure 4.3: An illustration of a generic flowchart. It contains the basic instructions for both the proposed and baseline algorithms.

present the proposed and baseline algorithms each of which are differed by the criteria they select the better host.

It has to be noted that some of the algorithms – Algorithm 1,2 and 4 – apply finding a host to a VM part of the generic flowchart twice: first to place a VM from the active hosts and second to place the VM from the inactive host lists (after turning them on) in case none of the active hosts fitting the VM.

Power Efficient First Fit Decreasing Algorithm

The Power Efficient First Fit Decreasing (PEFFD) algorithm as shown in Algorithm 1, first gets the VM list to be migrated and the host list to be allocated. For each VM sorted by decreasing resource demand (line 1 in Algorithm 1), the algorithm first tries to find a host that fits it and is the best (line 5-13). A host fits for a VM if it has enough resource for the VM (line 6). The best host is then determined by iteratively replacing *allocatedHost* with a better host (line 7-11). In PEFFD, a host is better than another host if its power efficiency, PE (see Equation 4.1), is greater than that of the other host and the lowest indexed one is chosen for a tie-breaking. Next, a VM and the best host are added to a *vmPlacement* map (line 15). If an active host is not found for a VM, then a host is searched from inactive host lists with same process (line 16-31). When all VMs are exhausted the *vmPlacement* map is returned.

Power Efficient Best Fit Decreasing Algorithm

The Power Efficient Best Fit Decreasing (PEBFD) algorithm shown in Algorithm 2 has a similarity to the PEFFD described above. The difference is PEBFD uses the best-fit rule instead of the first-fit rule. In PEBFD, a host is better than another host if its power efficiency, PE, is greater than that of the other host. In case the two hosts have the same PE, then the one that has lower available CPU is chosen (line 12-16 in Algorithm 2).

Medium Fit Power Efficient Decreasing Algorithm

The Medium Fit Power Efficient Decreasing (MFPED) algorithm is a modified form of MFD heuristic (see Section 4.3) in such a way that it will favor a higher PE in case of a tie. In MFPED a host is favored when its distance from the desired level, as defined in (4.2), is a minimum and in case two hosts have the same distance from the desired level, the one that has higher PE is chosen. The MFPED is listed in Algorithm 3.

The alternative combination is to favor the highest power-efficient (PE) host and in case of a tie to apply the MF rule to produce Power Efficient Medium Fit Decreasing (PEMFD).

Algorithm 1: Power Efficiency First Fit Decreasing (PEFFD)

Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement

```
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3      bestPowerEfficiency ← MIN ;          // MIN is the minimum number
4      allocatedHost ← NULL;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
8              if powerEfficiency > bestPowerEfficiency then
9                  allocatedHost ← host ;
10                 bestPowerEfficiency ← powerEfficiency ;
11             end
12         end
13     end
14     if allocatedHost ≠ NULL then
15         add (allocatedHost, vm) to vmPlacement ;
16     else
           // assign allocatedHost from inactive hosts
17         bestPowerEfficiency ← MIN;
18         allocatedHost ← NULL;
19         foreach host in inactiveHostList do
20             if host has enough resource for vm then
21                 powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
22                 if powerEfficiency > bestPowerEfficiency then
23                     allocatedHost ← host ;
24                     bestPowerEfficiency ← powerEfficiency ;
25                 end
26             end
27         end
28         if allocatedHost ≠ NULL then
29             add (allocatedHost, vm) to vmPlacement ;
30         end
31     end
32 end
```

Result: vmPlacement

Algorithm 2: Power Efficiency Best Fit Decreasing (PEBFD)

Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement

```
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3      bestPowerEfficiency ← MIN;
4      allocatedHost ← NULL;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
8              if powerEfficiency > bestPowerEfficiency then
9                  allocatedHost ← host ;
10                 bestPowerEfficiency ← powerEfficiency ;
11             else
12                 if powerEfficiency == bestPowerEfficiency then
13                     if getAvailableCPU(host) < getAvailableCPU(allocatedHost)
                        then
14                         allocatedHost ← host ;
15                     end
16                 end
17             end
18         end
19     end
20     if allocatedHost ≠ NULL then
21         add (allocatedHost, vm) to vmPlacement ;
22     else
           // assign allocatedHost from inactive hosts
23         bestPowerEfficiency ← MIN;
24         allocatedHost ← NULL;
25         foreach host in inactiveHostList do
26             if host has enough resource for vm then
27                 powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
28                 if powerEfficiency > bestPowerEfficiency then
29                     allocatedHost ← host ;
30                     bestPowerEfficiency ← powerEfficiency ;
31                 else
32                     if powerEfficiency == bestPowerEfficiency then
33                         if getAvailableCPU(host) <
                            getAvailableCPU(allocatedHost) then
34                             allocatedHost ← host ;
35                         end
36                     end
37                 end
38             end
39         end
40         if allocatedHost ≠ NULL then
41             add (allocatedHost, vm) to vmPlacement ;
42         end
43     end
44 end
```

Result: vmPlacement

However, it is the MF rule that is defined to improve SLA and that has to come first as in MFPED.

Algorithm 3: Medium Fit Power Efficient Decreasing (MFPED)

```
   Input: hostList, vmList
   Output: vmPlacement
 1 L_D ← 0.6 ;                                  // The desired level is set to 0.6
 2 sort vmList in the order of decreasing CPU utilization ;
 3 foreach vm in vmList do
 4 |   minDiff ← MAX ;                          // MAX is the maximum number
 5 |   allocatedHost ← NULL;
 6 |   foreach host in hostList do
 7 |   |   if host is active and has enough resource for vm then
 8 |   |   |   diff ← |getUtilization(host) − L_D| ;
 9 |   |   |   if diff < minDiff then
10 |   |   |   |   allocatedHost ← host ;
11 |   |   |   |   minDiff ← diff ;
12 |   |   |   else
13 |   |   |   |   if diff == minDiff then
   |   |   |   |   |   // PE(.)  is the power efficiency of a host
14 |   |   |   |   |   if PE(host) > PE(allocatedHost) then
15 |   |   |   |   |   |   allocatedHost ← host ;
16 |   |   |   |   |   end
17 |   |   |   |   end
18 |   |   |   end
19 |   |   end
20 |   end
21 |   if allocatedHost ≠ NULL then
22 |   |   add (allocatedHost, vm) to vmPlacement ;
23 |   end
24 end
   Result: vmPlacement
```

Chapter 5

Simulation and Evaluation

To evaluate the performance a VM placement algorithm, we conducted the following experiment in the simulation tool, CloudSim: i) a cloud data-center scenario is created, ii) the VM placement algorithm is implemented (in CloudSim) and specified as the current VM placement policy for the data-center, iii) workloads are specified and simulation start to run, and iv) performance metrics for measuring energy efficiency, quality degradation, and VM migrations are collected. The four procedures are repeated using baseline algorithms (see Appendix A) for comparison.

In the rest of this chapter, we first describe the simulation tool, CloudSim and present the cloud scenarios: data-center scenarios and workload traces. Next, we introduce the evaluation metrics used for measuring performance of proposed algorithms. Finally, we present evaluation results and conclude the chapter with summary and remarks.

5.1 CloudSim Simulator

CloudSim is an open-source toolkit for modeling and simulating cloud environment and evaluation of resource allocation algorithms [28]. It is widely used by research works in industry (such as HP Labs) and in academia [27, 25, 24, 26]. It offers the following novel features: (i) support for modeling and simulation of large-scale cloud computing environments, (ii) a platform for modeling clouds, service brokers, and resource allocation policies, and (iii) support for simulation of network connections among the simulated system.

CloudSim contains Java classes for modeling the different components of a cloud including classes for data-center, host, and virtual machine (see Figure 5.1). Users of the simulator customize it by extending the classes and overriding the methods.

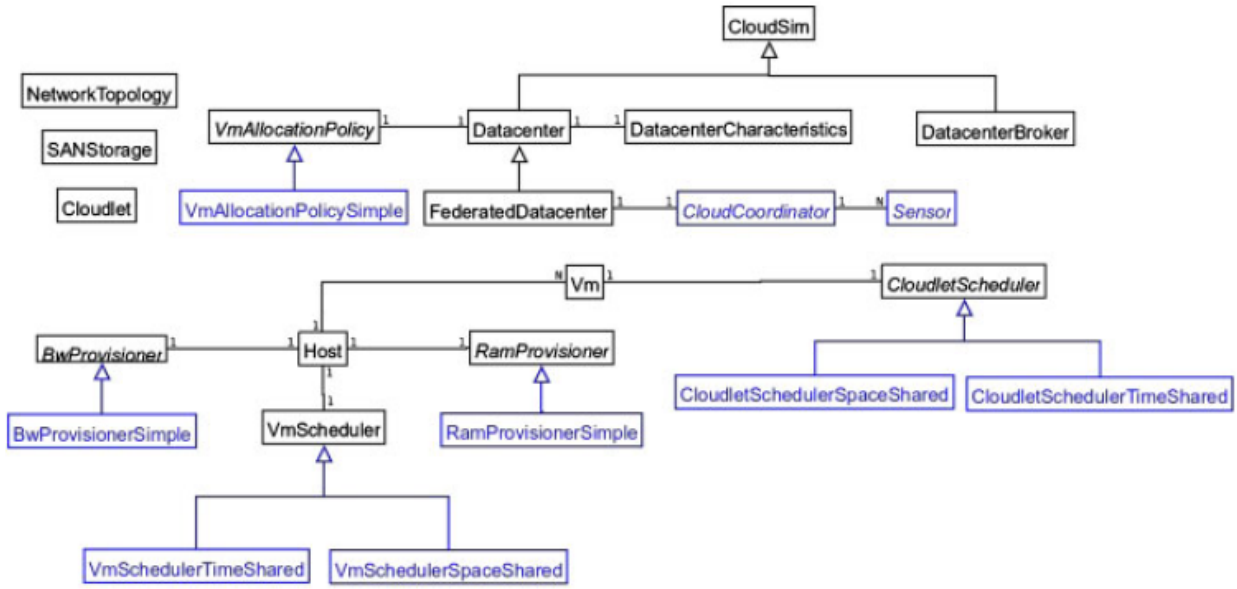


Figure 5.1: CloudSim class design diagram [28]

CloudSim classes

The following extract from Calheiros et al. (2011) [28] gives a brief description of CloudSim main classes..

CloudSim CloudSim is the main class responsible for managing event queues and controlling step-by-step (sequential) execution of simulation events.

Datacenter Datacenter encapsulates a set of compute hosts and implements a set of policies for allocating processor, memory, and storage devices to hosts and VMs.

DatacenterBroker This class models a broker that negotiates between SaaS and cloud providers.

DatacenterCharacteristics This class contains configuration information of data-center resources.

Host This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (to represent a multi-core machine), an allocation of policy for sharing the processing power among VMs.

Vm This class models a virtual machine(VM), which is managed and hosted by a Cloud host component. Every VM component has access to a component that stores the following characteristics related to a VM: accessible memory, processor, storage size, and the VM's internal provisioning policy that is extended from an abstract component called the CloudletScheduler.

VmAllocationPolicy This abstract class represents a provisioning policy that a VM Monitor utilizes for allocating VMs to hosts. The main function of the VmAllocationPolicy is to select the available host in a data center that meets the processor, memory, and storage requirement for a VM deployment.

VmScheduler This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processor cores to VMs. The functionalities of this class can easily be overridden to accommodate application-specific processor sharing policies.

Cloudlet This class models the Cloud-based application services. It has a pre-assigned instruction length and data transfer overhead.

CloudletScheduler This abstract class is extended by the implementation of different policies that determine the share of processing power among Cloudlets in a VM.

NetworkTopology This class contains the information for inducing network behavior (latencies) in the simulation.

Consolidation Framework in CloudSim

The consolidation framework of OpenStack Neat is included in the CloudSim in the form of power packages [27]. All algorithms except MBFD are included. For the VM placement host selection instead of MBFD, Power Aware Best-Fit Decreasing (PABFD) is included. Power packages are implemented by extending the cloud entity classes to be power aware. The list includes PowerDatacenter, PowerHost, PowerVM, PowerVMAllocation and PowerVMSelection policy. In addition it includes a power-model package that simulates real server's power model and a class for host utilization history that will be used by overload estimation algorithms. The description for some of the important classes is given in the following list.

PowerDatacenter PowerDatacenter is a class that enables simulation of power-aware data centers. It also updates processing of each cloudlet running in data-center and initiate optimization of VM allocation when the resource utilization of cloudlets changes.

PowerHost PowerHost class enables simulation of power-aware hosts. Besides inheriting the methods in Host class, PowerHost adds methods for setting power model of a host and getting power utilization at a given load.

PowerVM The class of a VM that stores its CPU utilization history. The history is used by VM allocation and selection policies.

PowerVmAllocationPolicyMigrationAbstract The class of an abstract power-aware VM allocation policy that dynamically optimizes the VM allocation using migration. Implementation of a VM placement algorithm can be achieved either by simply replacing the *findHostForVm* method of this class or by overbidding the method in an extended class.

PowerVmSelectionPolicy The class of an abstract VM selection policy for VM migration. This class can be extended to implement custom VM selection policy.

5.2 Cloud Scenarios

There cloud data-center scenarios are configured for evaluation: Default, Heterogeneous and Homogeneous. Each scenario contains host configuration, VM types and workloads that run in the data-center. The configuration of the three scenarios are described in the following.

Default-scenario

The first scenario, shown in Table 5.1, adopts the data-center setup of Beloglazov et al. [27] which is included in CloudSim. It has a cloud environment of 800 hosts from two server models (400 hosts from each server type) and four types of VMs. For the VM instances, their CPU capacity is given in millions of instruction per second (MIPS). The number of VMs and their workloads are generated from real clouds workload traces. The traces are from PlanetLab cloud [27] and Bitbrains cloud service provider [38, 39].

For overload prediction the local regression policy which is available in the CloudSim simulator is used. At each optimization step, the minimum loaded host is chosen for its VMs to be migrated. If the remaining hosts have enough resource to handle the VMs, the host will be turned off for saving energy.

Host types

In the *Default-scenario* two types of hosts are configured: (i) HpProLiantMl110G4 which has dual-core processors @1800 MHZ and 4GB RAM , and (ii) HpProLiantMl110G5 which has dual-core processors @2660 MHZ and 4GB RAM. The power model of those hosts are presented in Figure 5.2.

Table 5.1: Default-scenario parameters and configurations. The data-center host, VM and PlanetLab traces are adopted from the default configuration in CloudSim.

Parameters	Configuration
Host types	HP ProLiant ML110 G4 (2 X 1800 MIPS) HP ProLiant ML110 G5 (2 X 2660 MIPS)
Number of hosts	800; 400 of each host type
VM types	2500 MIPS 2000 MIPS 1500 MIPS 1000 MIPS
Workloads	PlanetLab (10 days of traces) Bitbrains (10 days of traces)
Overload decision	Local regression
Underload decision	The minimum loaded host

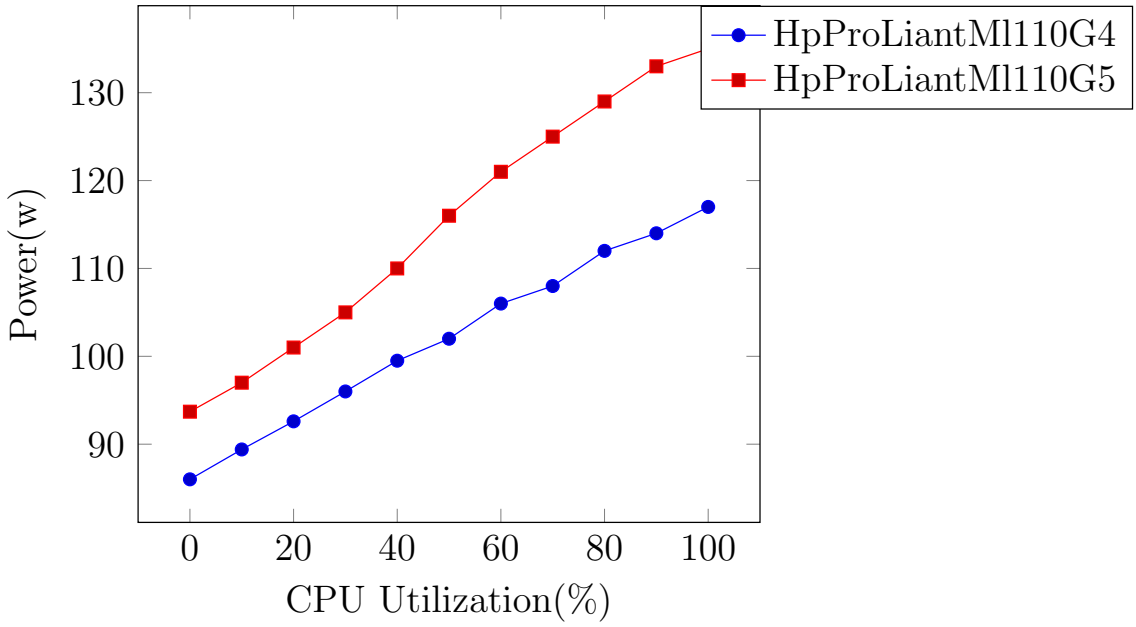


Figure 5.2: Power models for hosts in Default-scenario. The power efficiency, PE is 32 and 39 for HpProLiantMl110G4 and HpProLiantMl110G5 servers, respectively.

VM types

The VM instances included in CloudSim simulator are equivalent to those provided by Amazon cloud provider [27]. The following four VM instance types are used in our experiment:

- High-CPU Medium Instance: 2.5 EC2 Compute Units, 0.85 GB RAM
- Extra Large Instance: 2 EC2 Compute Units, 3.75 GB
- Small Instance: 1 EC2 Compute Unit, 1.7 GB RAM

- Micro Instance: 0.5 EC2 Compute Unit, 0.633 GB RAM

Heterogeneous-scenario

In this scenario the number of host types is increased to four by adding two quad-core IBM server models: IBM Xeon X3470 (4 X 2933 MIPS) and IBM Xeon X3480 (4 X 3067 MIPS). To make a comparable computational power as that of the Default-scenario, the number of hosts is reduced to 560 (140 of each server type). The CPU capacity and power model of all four hosts are shown in Table 5.2. The addition of two server types creates more heterogeneity in the cloud.

Table 5.2: Heterogeneous-scenario host configuration. PE is the power efficiency of hosts

Parameters	Configuration	PE
Host types	HP ProLiant ML110 G4 (2 X 1800 MIPS)	32
	HP ProLiant ML110 G5 (2 X 2660 MIPS)	39
	IBM Xeon X3470 (4 X 2933 MIPS)	104
	IBM Xeon X3480 (4 X 3067 MIPS)	108
Number of hosts	560, 140 of each type	

Homogeneous-scenario

In this scenario only one type of host is defined in the data-center. The setup differs from Default-scenario (Table 5.1) by the host type which in this case is only the HP ProLiant ML110 G5 server. In this scenario the power efficiency, PE, of all hosts are equal. Thus, performance improvement with respect to energy consumption is not expected from the proposed algorithms.

Workload traces

The experiment is performed on workload traces collected from real clouds: PlanetLab and Bitbrains. The PlanetLab is a cloud of global research network and the traces are collected from a monitoring system called CoMon [40]. The data contains the percentage of CPU utilization by more than a thousand VMs from servers located at more than 500 places around the world. It is collected during 10 randomly selected days in March and April 2010 [27]. The dataset is organized as one folder per day and a file in a folder contains one day CPU utilization of a VM sampled every 5 minutes. The statistical characteristics of the dataset is shown in Table 5.3. This workload is available with CloudSim simulator.

Table 5.3: Statistical characteristics of PlanetLab workloads traces. The percentages are relative to the configured CPU capacity of VMs.

Date	Number of VMs	Mean-load(%)	St.dev.(%)
03/03/2011	1052	12.31	17.09
06/03/2011	898	11.44	16.83
09/03/2011	1061	10.70	15.57
22/03/2011	1516	9.26	12.78
25/03/2011	1078	10.56	14.14
03/04/2011	1463	12.39	16.55
09/04/2011	1358	11.12	15.09
11/04/2011	1233	11.56	15.07
12/04/2011	1054	11.54	15.15
20/04/2011	1033	10.43	15.21

Bitbrains is a cloud service provider that specializes in managed hosting and business computation for enterprises [39]. The dataset collected contains resource utilization by 1,750 VMs from a distributed data-center and is available online from the Grid Workloads Archive [38]. It is organized into two folders: fastStorage traces consists of 1,250 VMs and Rnd traces consists of 500 VMs. In this work, we used the fastStorage traces. The dataset is organized as one file per a VM, each file containing 30 days of data sampled every 5 minutes.

To reuse the same utilization-model as that of PlanetLab available in CloudSim (*UtilizationModelPlanetLabInMemory* class), we have converted the datasets of Bitbrains to the format of PlanetLab datasets. The first 10 days of the converted datasets that are used in our experiment and their statistical characteristics are shown in Table 5.4.

Table 5.4: Statistical characteristics of Bitbrains workloads traces. The percentages are relative to the configured CPU capacity of VMs.

Date	Number of VMs	Mean-load(%)	St.dev.(%)
01/08/2013	1238	11.21	26.33
02/08/2013	1237	7.60	17.52
03/08/2013	1234	5.10	13.16
04/08/2013	1233	8.48	21.11
05/08/2013	1232	9.43	21.67
06/08/2013	1231	8.63	23.19
07/08/2013	1218	7.73	17.49
08/08/2013	1209	10.78	24.07
09/08/2013	1207	7.06	16.93
10/08/2013	1205	8.64	21.62

VM lists and workload assignments

The number of VMs are determined by the number of files in a workload folder. Workload folders are organized by date as shown in Table 5.3 and Table 5.4 for both PlanetLab and Bitbrains traces. For example, for workload folder at date 03/03/2011 of PlanetLab traces, 1052 number of VMs are created and that constitutes the VM lists for one particular experiment. The VM lists so created will have CPU capacity from the four types of VMs defined in Table 5.1. The first one-fourth of the VMs will have capacity of VM types I and so on. Each VM in the the VM lists is then assigned one of the workload files in the folder that determines its CPU utilization pattern for one day.

The initial placement of the VM lists to host lists is determined by the specified capacity of the VMs from the four types defined in Table 5.1 and the rule of the working VM placement algorithm. After initial placement, the optimization phase follows according to the OpenStack Neat framework which includes one of the VM placement algorithms for host selection. The resource demand of a VM, after initial placement, is read from the corresponding workload files.

5.3 Evaluation Metrics

We adopt the evaluation metrics proposed by Beleglazov et al. in [20]. The three main metrics are those that measure energy efficiency, SLA violation and number of VM migrations. Energy efficiency is measured with the total data-center energy consumption in kwh (kilowatt hour),

$$Energy_C \equiv \text{data-center energy consumption per day} \tag{5.1}$$

SLA violation is mostly due to resource overloading and is measured by aggregate Overload Time Fraction (OTF) and is defined as:

$$OTF \equiv \frac{1}{N}\sum_{i=1}^{N}\frac{T_{o_i}}{T_{a_i}} \tag{5.2}$$

Where,

- N is the number of hosts
- T_{o_i} is the total time during which the host i has experienced the utilization of 100% leading to an SLA violation
- T_{a_i} is the total time host i being in the active state (serving VMs)

The reasoning behind the OTF metric is the observation that if a host serving applications is experiencing the 100% utilization, the performance of the applications is bounded by the host's capacity; therefore, VMs are not being provided with the required performance level. In CloudSim simulation, T_{o_i} is counted whenever the CPU capacity requested exceeds the available capacity.

A VM migration causes overhead on a network as well as SLA violation and the associated metric is denoted by *#VM migration*.

$$\#VM\ migration \equiv \text{The number of VM migration in data-center per day} \tag{5.3}$$

In all defined metrics the lower the metric value is, the better the performance of the algorithm under consideration.

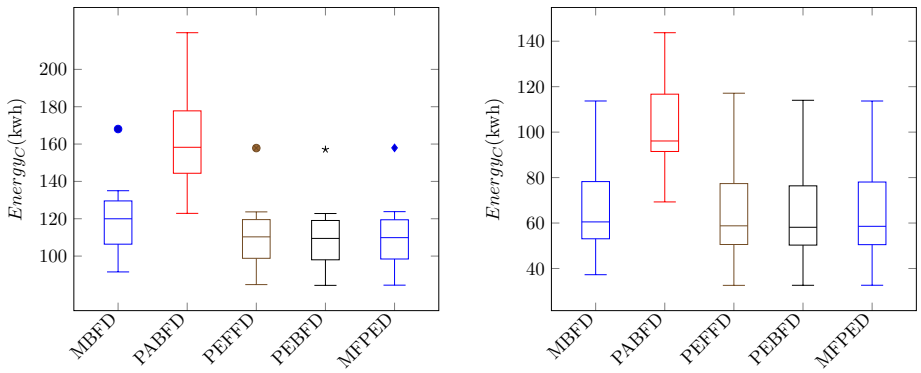
5.4 Results and Discussion

The results of the experiment for each scenario are discussed in the following subsections.

Performance of algorithms in the Default-scenario

In the Default-scenario all proposed algorithms deliver lower energy consumption, reduced SLA violation and VM migrations compared with baseline algorithms. Comparison by energy efficiency of the proposed algorithms against the baselines MBFD and PABFD are shown in Figure 5.3. From the box plots of Figure 5.3.a, we observe that PEBFD resulted in the lowest median energy consumption of 109.5 kwh in case of PlanetLab workload traces. It is followed by a closer results of MFPED and PEFFD with values 109.9 kwh and 110.3 kwh, respectively. The highest energy consumption has resulted from the baseline algorithm PABFD with a median value of 158.3 kwh followed by MBFD at 120 kwh. The improvement of proposed algorithms over baseline MBFD is 8 - 9%. In case of Bitbrains workload traces shown in Figure 5.3.b, the order of performance of the algorithms has not changed. The magnitude of the difference in energy consumption, however, shows significant change. For example, the improvement by the proposed algorithms in this case is 2 - 3% over the baseline MBFD.

Table 5.5 summarizes the average performance of the algorithms in the Default-scenario with respect to all defined metrics of Section 5.3: energy consumption, overload time fraction (OTF) and #VM migrations. In the table, the best values are highlighted in boldface. We observe that the proposed algorithms outperform the baseline algorithms in all performance metrics. The performance difference between the proposed algorithms is



(a) Results using PlanetLab traces (b) Results using Bitbrains traces

Figure 5.3: **Comparison by energy efficiency of algorithms in the Default-scenario.** Total energy consumption in data-center with two types of dual-core HP ProLiant servers. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones.

negligible($< 1\%$) with respect to average energy consumption while with respect to both OTF and #VM migrations, MFPED has the best performance. Compared with MBFD, MFPED improves OTF and #VM migrations by 32% and 15% respectively while using PlanetLab traces. Using Bitbrians traces, MFPED improves OTF and #VM migrations by 64% and 18%, respectively over MBFD.

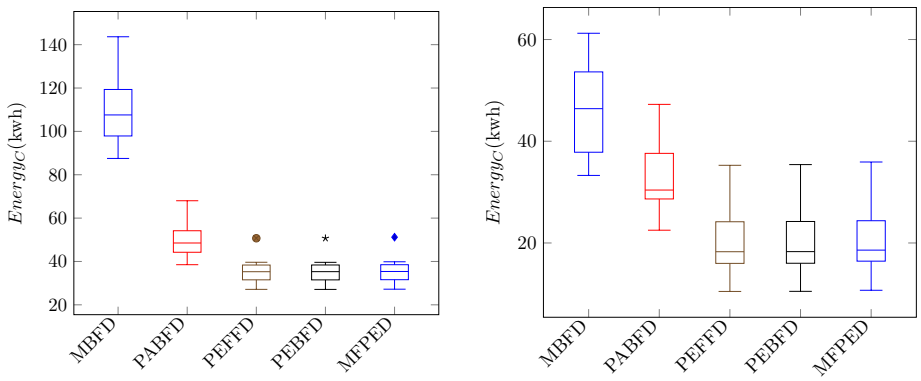
Table 5.5: Average performance of algorithms in the Default-scenario. The best values for each metrics defined in (5.1)-(5.3) are highlighted.

	Results using PlanetLab traces			Results using Bitbrians traces		
	$Energy_C$(kwh)	OTF(%)	#VM migration	$Energy_C$(kwh)	OTF(%)	#VM migration
MBFD	120.32	5.32	12919	67.49	6.82	8787
PABFD	161.87	6.21	28175	103.777	5.97	19808
PEFFD	111.13	4.06	12477	65.90	2.63	7612
PEBFD	**110.41**	4.21	11819	**65.17**	3.45	8527
MFPED	110.93	**3.62**	**10975**	65.57	**2.44**	**7216**

Thus we conclude that, in case of the Default-scenario, the proposed algorithms improve all metrics (energy consumption, SLA violation and number of VM migrations) irrespective of the workload traces considered.

Performance of algorithms in the Heterogeneous-scenario

In the Heterogeneous-scenario all proposed algorithms give lower energy consumption, reduced SLA violation and VM migrations compared with baseline algorithms. As shown in Figure 5.4 the proposed algorithms resulted in higher energy consumption improvement over baseline, MBFD, than the improvement in case of Default-scenario (on average 64%



(a) Results using PlanetLab traces (b) Results using Bitbrains traces

Figure 5.4: **Comparison by energy efficiency of algorithms in the Heterogeneous-scenario.** Total energy consumption in data-center with four types of servers from dual-core HP ProLiant and quad-core IBM Xeon. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones.

vs. 6%). The result shows that the power efficiency, PE, as defined in (4.1) is an important energy efficiency factor. From the box plots of Figure 5.4.a, we observe that all proposed algorithms resulted in a median energy consumption of about 35.3 kwh using PlanetLab workload traces. The highest energy consumption has resulted from the baseline algorithm MBFD with a median value of 107.6 kwh followed by PABFD, 48.5 kwh. The improvement of proposed algorithms over baseline, MBFD, is around 67%. In case of Bitbrains workload traces shown in Figure 5.4.b, the lowest median energy consumption has resulted from both PEFFD and PEBFD with a value of 18.3 kwh. The result is followed by a very close value of MFPED, 18.6 kwh. The improvement of proposed algorithms, in this case, over the baseline MBFD is about 60%.

Table 5.6 presents the average performance of the algorithms with respect to all defined metrics for Heterogeneous-scenario. The performance difference between proposed algorithms is negligible ($< 0.5\%$) with respect to energy consumption. With respect to OTF and #VM migrations, MFPED has the lowest value followed by PEFFD. The improvement of MFPED over the baseline MBFD is 68% for OTF and 46% for #VM migrations. In case of Bitbrains traces too, the proposed algorithms improve energy consumption against both baseline algorithms. Also, energy performance differences among the proposed algorithms are negligible. The MFPED algorithm improves OTF and #VM migration by 78% and 40% respectively, against MBFD.

Thus, in Heterogeneous-scenario, the proposed algorithms improve all metrics (energy consumption, SLA violation and number of VM migration) by a greater amount than the improvement in case of the Default-scenario.

Table 5.6: Average performance of algorithms in the Heterogeneous-scenario. The best values for each metrics defined in (5.1)-(5.3) are highlighted.

	Results using PlanetLab traces			Results using Bitbrians traces		
	$Energy_C$(kwh)	OTF(%)	#VM migration	$Energy_C$(kwh)	OTF(%)	#VM migration
MBFD	109.31	5.51	12524	46.11	6.56	8360
PABFD	49.66	6.04	18160	33.29	6.45	17826
PEFFD	**35.58**	2.06	7452	**20.56**	1.82	5652
PEBFD	**35.58**	2.18	7853	20.59	2.07	5903
MFPED	35.74	**1.74**	**6731**	20.88	**1.48**	**5041**

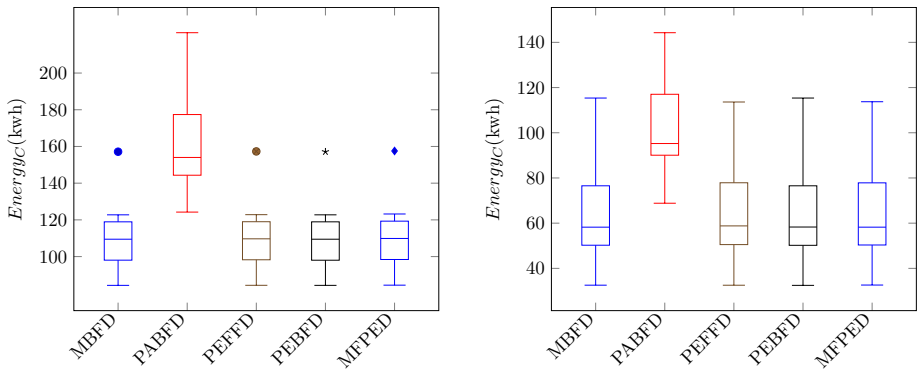
Performance of algorithms in the Homogeneous-scenario

There is no average energy saving by the proposed algorithms against baseline MBFD in case of Homogeneous-scenario. The benefit of the proposed algorithms, in this case, is on reducing overload time fraction and number of VM migrations. In Homogeneous-scenario, as the power efficiency of all hosts is the same, the only power saving VM placement method is to minimize the number of hosts. Thus, we do not expect an improvement in energy efficiency from proposed algorithms. As shown in Figure 5.5, both BFD based algorithms, MBFD and PEBFD, resulted in nearly equal median energy consumption of 109.5 kwh and 58.2 kwh for PlanetLab and Bitbrains traces, respectively. The worst energy efficiency (highest consumption) has resulted from PABFD in both workload traces.

Table 5.7 presents the average result of the experiment in Homogeneous-scenario using PlanetLab traces. There is no average energy saving by the proposed algorithms against MBFD. There is, however, improvement in reducing overload time fraction and number of VM migrations. Compared to MBFD, MFPED improves OTF and #VM migrations by 19% and 9%, respectively, in PlanetLab traces. Similarly, in case of Bitbrians traces, the two BFD based algorithms, MBFD and PEBFD, resulted in almost equal energy consumption. Using Bitbrians traces, MFPED improves OTF and #VM migrations by 42% and 14%, respectively over MBFD. We also note that the worst algorithm with respect to all three metrics in both workload traces is the PABFD .

Table 5.7: Average performance of algorithms in Homogeneous-scenario. The best values for each metrics defined in (5.1)-(5.3) are highlighted.

	Results using PlanetLab traces			Results using Bitbrians traces		
	$Energy_C$(kwh)	OTF(%)	#VM migration	$Energy_C$(kwh)	OTF(%)	#VM migration
MBFD	**110.40**	4.26	11898	65.29	3.19	8540
PABFD	161.43	6.26	27980	103.70	6.00	19625
PEFFD	110.57	4.00	12035	65.59	2.64	7578
PEBFD	**110.40**	4.27	11909	**65.28**	3.16	8408
MFPED	110.78	**3.60**	**10914**	65.39	**2.24**	**7460**



(a) Results using PlanetLab traces (b) Results using Bitbrains traces

Figure 5.5: **Comparison by energy efficiency of algorithms in the Homogeneous-scenario.** Total energy consumption in data-center with one type of server: HP ProLiant ML110 G5. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones.

Tests of Significance

Once sample data has been collected through an observation or experiment, statistical inference allows to assess evidence in favor of some claim about the population from which the sample has been drawn. The test used to support or reject a claim based on a sample data is known as tests of significance [41]. Some test of significance methods such as *Paired t-Test* requires the populations of interest to be normally distributed. We have conducted normality test on the distribution of the experiment results using Kolmogorov-Smirnov method [42]. The result shows that the observation in our experiment are not normally distributed. Thus, we conducted the statistical significance test using the Wilcoxon signed-rank test which have no constraint on the distribution of the data being tested [43].

In this subsection two tests are presented: (i) comparing the energy consumption of a data-center under proposed algorithms with the value under MBFD (Table 5.8), (ii) comparing OTF and #VM migrations value of MFPED algorithm with that of MBFD (Table 5.9). Like any statistical significance test two hypothesis are defined: the null hypothesis and an alternative hypothesis. In our case, the null hypothesis states that *the proposed algorithm has no performance improvement when compared to the baseline algorithm.* The alternative hypothesis is the hypothesis we want to approve and it states that *the proposed algorithm gives a better performance than the baseline algorithm.* We want to reject the null hypothesis with a significance level of $\alpha = 0.01$ for all tests.

The P-value in statistical significance indicates the probability of the null hypothesis. A P-value less than α implies the null hypothesis can be rejected or the alternative hypothesis can be accepted with significance level of α. In our case the results of the

Table 5.8: Statistical test for the significance of energy saving by the proposed algorithms

Comparison by energy consumption,E	$P - value$
E(PEFFD) < E(MBFD)	9.8×10^{-4}
E(PEBFD) < E(MBFD)	9.8×10^{-4}
E(MFPED) < E(MBFD)	9.8×10^{-4}

Table 5.9: Statistical test for the significance of OTF and #VM migrations improvement by the proposed MFPED algorithm

Comparisons by OTF & #VM migrations(Mig)	$P - value$
OTF(MFPED) < OTF(MBFD)	9.8×10^{-4}
Mig(MFPED) < Mig(MBFD)	9.8×10^{-4}

P-values are less than the significant level $\alpha = 0.01$ in all tests ((see Tables 5.8 and 5.9). Which means that the improvement in all defined metrics are significant with confidence level of at least 99%.

The above tests are for the default cloud data-center scenario. Statistical significance tests for the improvements in case of Heterogeneous-scenario give similar results to the above (see Appendix C).

5.5 Summary of Results

To summarize, in both default and heterogeneous power model configuration the proposed algorithms provide a better performance than the baseline algorithms. The performance difference between proposed algorithms is negligible with respect to energy consumption. With respect to the other metrics such as overload time fraction (OTF) and #VM migration the MFPED performs the best. Statistical tests show that the improvement in total energy consumption, SLA violation, and VM migrations are highly significant.

Another important observation is that the relative performance of the algorithms is independent of the workload data used. It means that the choice of which VM placement algorithm to use does not depend on the type of service running in the cloud.

Finally, we observe that the data-center host types affect the relative performance of the algorithms. For example, in the homogeneous setup all algorithms except PABFD have about the same energy consumption; the PABFD and the proposed algorithms resulted in a relatively higher energy efficiency when there is big difference among the power utilization efficiency of hosts.

Chapter 6

Conclusions

Energy efficient operation is one of the main concern in today's information systems. In cloud computing, consolidation techniques enables such operation by migrating VMs to the fewest possible servers and turning off or putting in sleep mode the rest of the servers. However, one of the most common and widely used open-source cloud management tool, OpenStack, lacks a consolidation framework in its resource scheduler. From state-of-the-art survey, we have found that OpenStack Neat is a suitable consolidation solution for OpenStack.

OpenStack Neat is an open-source consolidation framework that can be seamlessly integrated to OpenStack. The framework has four decision components: (i) a component that decides whether a host is overloaded, (ii) a component that decides when a host is underloaded, (iii) a component for VM selection for migration, and (iv) a host selection for VM placement. For the VM placement the modified best-fit algorithm (MBFD) is used. In this thesis, after identifying some limitation of MBFD, we set objective to propose an enhanced VM placement algorithms.

To improve the energy efficiency, we propose VM placement algorithms based on both bin-packing heuristics and servers' power efficiency. Two algorithms that are developed using such techniques are the power-efficient first-fit decreasing (PEFFD) and the power-efficient best-fit decreasing (PEBFD) algorithms.

To evaluate the performance of the proposed algorithms we have conducted experiments using CloudSim on three cloud data-center scenarios: homogeneous, heterogeneous and default. The workloads that run in the data-center scenarios are generated from traces of PlanetLab and Bitbrains clouds. The results of the experiment show up-to 67% improvement in energy consumption.

Moreover, to reduce SLA violation and VM migrations caused by best-fit based algorithms, we defined a new Bin-packing rule called a medium-fit. By modifying the medium-

fit rule with power-efficiency we developed what we call the medium-fit power-efficient decreasing (MFPED) algorithm. Compared with other VM placement algorithms, the MFPED deliver a minimum SLA violation and number of VM migrations. It improves SLA violation and number of VM migrations over MBFD by up-to 78% and 46%, respectively, depending on the cloud scenario.

All improvements in energy consumption, SLA violation and amount of VM migrations are statistically significant with $\alpha = 0.01$ significance level.

6.1 Contributions

The main contribution of this thesis work to the state of the art VM placement algorithms are the following.

- A new bin-packing heuristic called Medium-Fit (MF) is defined which provides a well-balanced efficiency between energy consumption and SLA assurance.
- An approach for modifying bin-packing heuristics based on the power efficiency of servers is proposed. The approach resulted in a more energy efficient algorithms than the bin-packing huerstics for a heterogeneous cloud.
- The proposed algorithms can be implemented with a minor addition to a cloud configuration database. The peak power of hosts is the only additional information necessary to incorporate the proposed algorithms in the implementation of OpenStack Neat framework.

6.2 Future Research Directions

The algorithms proposed in this thesis allocate resource based on the current available resource of hosts and resource demands of VMs. Thus, a possible limitations of the proposed VM algorithms may arise from the inability to accurately predicting the future resource demand. The lack of future resource demand prediction capability may lead to the loss in the performance of SLA and VM migrations. In future work we plan to adopt the proposed VM placement algorithms to accommodate future resource demands of VMs and evaluate their consolidation performance.

Furthermore, the proposed algorithms have the Bin-packing model as the core of the VM placement rule. The algorithms converge very fast and their computation complexity is quadratic in the number of virtual machines, the same as the baseline algorithms. However, there are other class of algorithms with higher complexity that can be used for

VM placement. One such class of algorithms are genetic algorithms. Thus, as a future work we plan to consider other class of algorithms for developing an energy and SLA efficient VM placement algorithms.

Considering the scope of the thesis work, it is all focused on computing elements of the data-center (or servers). Computing devices are not the only elements in data-center that consume energy. Other elements like network equipments or air conditioning units also contribute to the total energy consumption. For the future work, we recommend to extend the scope of the work to include network devices and traffic effects in the consolidation problem.

We also recommend modification of the proposed solution for specific cases of cloud applications. For example, in cases where there is a high communication need between groups of virtual machines, the VM selection for migration should be aware of the bandwidth demand of VMs for its decision. Customizing the VM placement algorithms for a specific cloud service provider need is also another possible research area.

References

[1] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, p. 50, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1721654.1721672

[2] L. C. Qi Zhang Raouf Boutaba, "Cloud computing: state-of-the art and research challenges," *Brazilian Comput. Soc.*, vol. 1, pp. 1–12, 2010.

[3] "Cloud computing environment and security challenges: A review," *International Journal of Advanced Computer Science and Applications*.

[4] J. G. Koomey, "Growth in data center electricity use 2005 to 2010," *Analytics Press Technical report*, Aug 2011.

[5] S. Ismaeel, R. Karim, and A. Miri, "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres," *Journal of Cloud Computing*, vol. 7, no. 1, p. 10, Jun 2018. [Online]. Available: https://doi.org/10.1186/s13677-018-0111-x

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," *in Proceedings of the 2Nd Conference on Symposium on Networked Systems Design and Implementation*, vol. 2, 2005.

[7] D. Meisner, B. Gold, and T. Wenisch, "Powernap: eliminating server idle power," *ACM SIGPLAN Notices*, vol. 44, pp. 205–216, 2007.

[8] M. Guazzone, C. Anglano, and M. Canonico, "Exploiting vm migration for the automated power and performance management of green cloud computing systems," *1st International Workshop on Energy Efficient Data Centers*, pp. 81–92, 2012.

[9] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," *2010 IEEE Netw. Oper. Manag. Symp. - NOMS 2010*, pp. 479–486, 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5488484/

[10] J. L. Berral, R. Gavaldà, and J. Torres, "Adaptive scheduling on power-aware managed data-centers using machine learning," *Proc. - 2011 12th IEEE/ACM Int. Conf. Grid Comput. Grid 2011*, pp. 66–73, 2011.

[11] J. Berral, R. Gavalda, and J. Torres, "Power-aware Multi-DataCenter Management using Machine Learning," *42nd Int. Conf. Parallel Process.*, 2013.

[12] F. F. Moghaddam, R. F. Moghaddam, and M. Cheriet, "Carbon-aware distributed cloud: multi-level grouping genetic algorithm," *Cluster Comput.*, vol. 18, no. 1, pp. 477–491, 2015.

[13] J. Zhang, L. Zhang, H. Huang, X. Wang, C. Gu, and Z. He, "A Unified Algorithm for Virtual Desktops Placement in Distributed Cloud Computing," *Math. Probl. Eng.*, 2016.

[14] D. Dong, P. Stack, H. Xiong, and J. P. Morrison, "Managing and unifying heterogeneous resources in cloud environments," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, INSTICC. SciTePress, 2017, pp. 143–150.

[15] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula," *Proc. - 2012 9th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2012*, pp. 2457–2461, may 2012.

[16] R. Kumar, N. Gupta, S. Charu, K. Jain, and S. K. Jangir, "Open Source Solution for Cloud Computing Platform Using OpenStack," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 5, pp. 89–98, may 2014.

[17] OpenStack. Openstack: Compute schedulers. Accessed 25 Jan 2018. [Online]. Available: https://docs.openstack.org/newton/config-reference/compute/schedulers.html

[18] Y. Huanle, S. Weifeng, and B. Tingting, "An openstack-based resource optimization scheduling framework," *International Symposium on Computational Intelligence and Design*, 2013.

[19] A. Beloglazov and R. Buyya. Openstack neat: Dynamic consolidation of virtual machines on openstack. Accessed 27 Jan 2018. [Online]. Available: http://openstack-neat.org/

[20] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, 2012.

[21] A. Ranjini and A. Sahayadhas, "A comparison study of various virtual machine

consolidation algorithms in cloud datacenter," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, 2017.

[22] M. Guazzone, "Power and performance management in cloud computing systems," Ph.D. Thesis, University of Torino, Computer Science Department, 2012.

[23] A. Beloglazov and R. Buyya, "Openstack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds," *Concurrency and Computation: Practice and Experience*, May 2014.

[24] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various VM placement strategies in CloudSim," *J. Cloud Comput.*, vol. 4, no. 1, pp. 1–21, 2015.

[25] Z. Á. Mann and M. Szabó, "Which is the best algorithm for virtual machine placement optimization?" *Concurr. Comput. Pract. Exp.*, vol. 29, no. 10, p. e4083, 2017. [Online]. Available: http://doi.wiley.com/10.1002/cpe.4083

[26] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing," *Proc. - 2015 IEEE 8th Int. Conf. Cloud Comput. CLOUD 2015*, pp. 381–388, jul 2015.

[27] 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," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[28] R. N. Calheiros, R. Ranjan, A. Beloglazov, a. F. D. R. Cesar, and R. Buyya, "CloudSim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource," *Softw. - Pract. Exp.*, vol. 41, no. 1, pp. 23–50, jan 2011. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/spe.995/pdf

[29] P. D. Leedy and J. E. Ormrod, *Practical Research: Planning and Design, 11th Ed.* Boston: Pearson, 2015.

[30] Z. Mann, "Allocation of virtual machines in cloud data centers: A survey of problem models and optimization algorithms," *ACM Computing Surveys*, vol. 48, 2015.

[31] E. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: Survey and classification," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 455–531.

[32] D. S. Johnson, "Near-optimal bin packing algorithms," Ph.D. Thesis, Massachusetts Institute of Technology, 1973.

[33] E. Coffman, M. Garey, and D. Johnson, "Approximation algorithms for bin backing:

a survey," in *Approximation Algorithms for NP-Hard Problems*. Springer, 1997, pp. 46–93.

[34] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms," *SIAM J. Comput.*, vol. 3, no. 4, pp. 299–325, 1974. [Online]. Available: http://epubs.siam.org/doi/10.1137/0203025

[35] M. Yue, "A simple proof of the inequality $ffd(l) < 11/9opt(l) + 1, \forall l$ for the ffd bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, 1991.

[36] Lei Shi, J. Furlong, and Runxin Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," *2013 IEEE Symp. Comput. Commun.*, pp. 000 009–000 015, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6754915/

[37] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers, Marvin V. Zelkowitz(ed.)*, vol. 82, 2011.

[38] S. Anoep, C. Dumitrescu, D. Epema, A. Iosup, M. Jan, H. Li, and L. Wolters. The Grid Workloads Archive: Bitbrains. Accessed 20 March 2018. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

[39] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," *Proc. - 2015 IEEE/ACM 15th Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2015*, pp. 465–474, 2015.

[40] K. S. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planet-lab," *ACM SIGOPS Operating Systems Review*, vol. 40, pp. 65–74, 2006.

[41] V. J. Easton and J. H. McColl, *Statistics Glossary v1.1*, 1997. [Online]. Available: http://www.stats.gla.ac.uk/steps/glossary/

[42] F. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, pp. 68–78, 1951.

[43] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers. 3rd Edition.*, 2003.

Appendices

A Baseline Algorithms

Two baseline algorithms: the MBFD, a VM placement for OpenStack Neat tool [23, 19], and the PABFD, a VM placement in CloudSim [20, 27], are given in Algorithm 4 and 5, respectively.

In MBFD (Algorithm 4), a host is better than another host if its available CPU resource is less than that of the other host. If the two hosts have same free CPU resource then minimum available RAM is used as a tiebreaker. For PABFD (Algorithm 5), a host is better than another if the estimated power for the current VM is smaller than the one estimated for the other host.

Algorithm 4: Modified Best Fit Decreasing (MBFD)

Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement

```
1  sort vmList in the order of decreasing average CPU utilization ;
2  foreach vm in vmList do
3      minCPU ← MAX;
4      allocatedHost ← NULL;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              cpu ← getAvailableCPU(host) ;
8              if cpu < minCPU then
9                  allocatedHost ← host ;
10                 minCPU ← cpu ;
11             else
12                 if cpu == minCPU and
                    getAvailableRAM(host) < getAvailableRAM(allocatedHost)
                    then
13                     allocatedHost ← host
14                 end
15             end
16         end
17     end
18     if allocatedHost ≠ NULL then
19         add (allocatedHost, vm) to vmPlacement ;
20     else
           // assign allocatedHost from inactive hosts
21         minCPU ← MAX;
22         allocatedHost ← NULL;
23         foreach host in inactiveHostList do
24             if host has enough resource for vm then
25                 cpu ← getAvailableCPU(host) ;
26                 if cpu < minCPU then
27                     allocatedHost ← host ;
28                     minCPU ← cpu ;
29                 else
30                     if cpu == minCPU and
                        getAvailableRAM(host) < getAvailableRAM(allocatedHost)
                        then
31                         allocatedHost ← host
32                     end
33                 end
34             end
35         end
36         if allocatedHost ≠ NULL then
37             add (allocatedHost, vm) to vmPlacement ;
38         end
39     end
40 end
```

Result: vmPlacement

Algorithm 5: Power Aware Best Fit Decreasing (PABFD)

Input: hostList, vmList
Output: vmPlacement

```
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3  |  minPower ← MAX;
4  |  allocatedHost ← NULL;
5  |  foreach host in hostList do
6  |  |  if host has enough resources for vm then
7  |  |  |  power ← estimatePower(host, vm) ;
8  |  |  |  if power < minPower then
9  |  |  |  |  allocatedHost ← host ;
10 |  |  |  |  minPower ← power ;
11 |  |  |  end
12 |  |  end
13 |  end
14 end
15 if allocatedHost ≠ NULL then
16 |  add (allocatedHost, vm) to vmPlacement ;
17 end
```

Result: vmPlacement

B Availability

All source codes and converted workload traces are publicly available from the following URL: https://github.com/fikrufeleke/vmplacement.

C Tests of Significance for the Case of Heterogeneous-scenario

The following lists show the energy consumption, E(.); the overload time fraction, OTF(.); and the number of migration, Mig(.) performance of the VM placement algorithms for 10 experiments in heterogeneous cloud data-center setup using PlanetLab traces:

E(MBFD): [121.8,87.52,97.55,111.43,98.91,143.65,120.2,116.74,103.75,91.57]

E(PEFFD): [37.02,27.59,31.08,37.65,32.87,50.71,39.63,38.59,33.53,27.13]

E(PEBFD): [37.02,27.6,31.07,37.62,32.81,50.72,39.6,38.63,33.58,27.11]

E(MFPED): [37.12,27.62,31.15,37.95,32.93,51.2,39.86,38.69,33.64,27.21]

OTF(MBFD): [4.13,5.02,7.95,5.5,4.31,4.76,4.69,4.78,5.55,8.4]

Table C.1: Statistical test for the significance of energy saving by the proposed algorithms in case of Heterogeneous-scenario

Comparison by energy consumption,E	$P - value$
E(PEFFD) < E(MBFD)	9.8×10^{-4}
E(PEBFD) < E(MBFD)	9.8×10^{-4}
E(MFPED) < E(MBFD)	9.8×10^{-4}

Table C.2: Statistical test for the significance of OTF and #VM migrations improvement by the proposed MFPED algorithm in case of Heterogeneous-scenario

Comparisons by OTF & #VM migrations(Mig)	$P - value$
OTF(MFPED) < OTF(MBFD)	9.8×10^{-4}
Mig(MFPED) < Mig(MBFD)	9.8×10^{-4}

OTF(MFPED): [1.4,1.24,2.09,1.82,1.66,1.61,1.62,1.58,1.9,2.45]

Mig(MBFD): [11587,8610,11706,14630,11820,15783,13866,13654,11501,12082]

Mig(MFPED): [5545,4439,6474,8090,6179,9365,7033,7274,6453,6458]

The following lists show the same metrics in case of Bitbrains traces:

E(MBFD): [47.72,39.85,37.17,55.26,45.09,48.76,33.28,56.21,36.53,61.23]

E(PEFFD): [15.91,24.19,10.47,16.72,24.12,35.27,19.84,16.22,15.74,27.08]

E(PEBFD): [15.94,24.27,10.49,16.81,24.09,35.41,19.79,16.23,15.74,27.08]

E(MFPED): [16.4,24.39,10.7,17.27,24.33,35.92,19.92,16.5,15.92,27.4]

OTF(MBFD): [9.34,2.59,5.41,7.21,9.17,1.03,4.16,13.11,7.94,5.64]

OTF(MFPED): [1.34,1.14,1.7,1.12,1.61,0.39,1.35,1.49,2.97,1.64]

Mig(MBFD): [8791,8000,8747,8726,10061,4785,6598,9069,9265,9558]

Mig(MFPED): [4709,4720,5113,4552,6293,2868,4642,4864,6117,6527]

The significance test results with $\alpha = 0.01$ for both workload traces are the same and shown in Table C.1 and C.2. This results show the significance of the proposed algorithms for Heterogeneous-scenario.